microservices patterns pdf github

microservices patterns pdf github is a popular search query for developers and architects seeking to understand and implement robust microservices architectures. This article provides a comprehensive guide to microservices patterns, drawing extensively from the rich resources available on GitHub, often presented in PDF formats. We will delve into the core concepts of microservices, explore essential design patterns for building scalable and resilient systems, discuss common challenges and their solutions, and highlight how GitHub serves as a crucial hub for accessing and sharing this invaluable knowledge. Whether you're migrating from a monolithic application or designing a new distributed system, understanding these patterns is paramount for success.

Table of Contents

- Deployment and Operational Patterns for Microservices
- Leveraging GitHub for Microservices Patterns
- Finding Microservices Patterns PDFs on GitHub
- Practical Implementation with GitHub Examples
- Common Challenges and Solutions in Microservices
- Future Trends in Microservices Patterns

Understanding Microservices Architecture

Microservices architecture is an architectural style that structures an application as a collection of small, independent, and loosely coupled services. Each service is built around a specific business capability, communicates via lightweight protocols, and can be deployed, scaled, and managed independently. This contrasts sharply with monolithic architectures where an entire application is built as a single, unified unit. The benefits of adopting microservices include increased agility, improved scalability, technology diversity, and enhanced fault isolation. However, the distributed nature of microservices introduces complexities that necessitate well-defined architectural patterns.

In a microservices setup, each service can have its own database, its own technology stack, and its own development team. This autonomy allows for faster development cycles and the ability to adopt new technologies more readily. The goal is to achieve a system that is easier to understand, develop, test, and deploy compared to large, complex monoliths. Understanding the foundational principles of microservices is the first step before diving into the specific patterns that govern their behavior and interaction.

The Importance of Microservices Patterns

Microservices patterns are reusable solutions to common problems encountered when designing, building, and operating microservices-based systems. They provide a common vocabulary and proven approaches for tackling challenges related to service decomposition, inter-service communication, data management, fault tolerance, and operational concerns. Without these patterns, teams might reinvent the wheel, leading to inconsistent designs, increased complexity, and a higher risk of failure. Patterns act as blueprints, guiding architects and developers towards robust and scalable solutions.

These patterns are born out of practical experience and collective wisdom within the software development community. They abstract away the intricate details of implementation, focusing on the underlying principles and benefits. By adhering to established patterns, development teams can build systems that are more predictable, maintainable, and resilient. The availability of these patterns in formats like PDFs, often hosted on platforms like GitHub, makes them accessible for study and reference.

Key Microservices Patterns: A Deep Dive

The landscape of microservices patterns is vast, covering various aspects of the architecture. These patterns are not prescriptive rules but rather guidelines that offer flexibility and adaptability to specific project needs. Understanding the intent and applicability of each pattern is crucial for making informed architectural decisions. We can broadly categorize these patterns based on the problem they aim to solve, such as how services are broken down, how they communicate, and how their data is managed.

The continuous evolution of microservices has led to the identification and refinement of numerous patterns. Mastering these patterns is essential for anyone involved in designing or managing distributed systems. The following sections will explore some of the most critical and widely adopted microservices patterns, often found documented in comprehensive PDFs and code examples on

GitHub.

Decomposition Patterns for Microservices

Decomposition is a fundamental aspect of microservices architecture, determining how a large application is broken down into smaller, manageable services. The way services are divided significantly impacts system cohesion, coupling, and maintainability. Incorrect decomposition can lead to overly chatty services, tight coupling between them, or services that are too large, negating the benefits of microservices.

Decomposition by Business Capability

This is a highly recommended pattern where each microservice is aligned with a specific business capability. For instance, an e-commerce application might have services for Order Management, Product Catalog, Customer Service, and Payment Processing. Each service encapsulates the logic and data related to its domain, promoting high cohesion within the service and low coupling between services.

Decomposition by Subdomain

Drawing from Domain-Driven Design (DDD), this pattern decomposes the system based on the different subdomains identified within the business domain. Each subdomain maps to a microservice, ensuring that the service boundaries align with the business's understanding of its own structure. This approach often leads to services that are well-defined and have clear responsibilities.

Decomposition by Use Case

While less common for the primary decomposition strategy, some teams might decompose by specific

use cases or user journeys. This can be useful for certain scenarios, but it often leads to services that are too fine-grained and can result in significant duplication of logic if not managed carefully. It's generally preferred to combine this with other decomposition strategies.

Integration Patterns for Microservices

Inter-service communication is a cornerstone of microservices. As services are distributed, they need robust mechanisms to exchange information and coordinate actions. Various patterns exist to manage these interactions, each with its own trade-offs regarding performance, reliability, and complexity. Choosing the right integration pattern is vital for a well-functioning distributed system.

API Gateway

An API Gateway acts as a single entry point for all client requests. It routes requests to the appropriate microservice, aggregates responses, and can handle cross-cutting concerns like authentication, authorization, rate limiting, and logging. This pattern simplifies client interactions and shields clients from the underlying service topology.

Choreography vs. Orchestration

These are two primary approaches to managing workflows involving multiple services. In choreography, services react to events emitted by other services without a central controller. In orchestration, a dedicated orchestrator service dictates the flow of execution, commanding other services to perform tasks. Choreography promotes looser coupling but can be harder to understand and debug. Orchestration offers more control but can become a bottleneck and a single point of failure.

Message Queues and Event Buses

Asynchronous communication using message queues (e.g., RabbitMQ, Kafka) or event buses allows services to communicate without direct coupling. Services publish messages or events, and other interested services subscribe to them. This decouples senders and receivers, improving resilience and scalability by enabling buffering and retries.

Observability Patterns for Microservices

In a distributed system, understanding what's happening within and between microservices is critical for troubleshooting, performance monitoring, and identifying issues. Observability patterns focus on making the system's internal state visible through logs, metrics, and traces.

Distributed Tracing

Distributed tracing allows you to follow a single request as it travels through multiple microservices. Each service adds context to the trace, creating a complete picture of the request's journey, including latency at each step. This is invaluable for pinpointing performance bottlenecks and errors.

Centralized Logging

Aggregating logs from all microservices into a central location (e.g., Elasticsearch, Splunk) makes it easier to search, analyze, and correlate log data across the entire system. This helps in debugging and understanding system behavior.

Application Performance Monitoring (APM)

APM tools provide real-time insights into application performance, including response times, error

rates, and resource utilization for individual services and the system as a whole. This enables proactive identification and resolution of performance issues.

Resilience Patterns for Microservices

Microservices, being distributed, are inherently susceptible to failures. Resilience patterns aim to ensure that the system continues to function, perhaps in a degraded mode, even when individual services fail. These patterns are crucial for building fault-tolerant systems.

Circuit Breaker

A circuit breaker pattern prevents an application from repeatedly trying to execute an operation that's likely to fail. After a certain number of failures, the circuit breaker "opens," and subsequent calls to that operation are immediately failed or return a fallback response, preventing cascading failures.

Bulkhead

Inspired by ship bulkheads, this pattern isolates elements of an application into pools so that if one fails, the others will continue to function. In microservices, this can mean isolating different types of requests or different groups of users to prevent one failing operation from impacting others.

Retry Pattern

When a service call fails due to transient network issues or temporary service unavailability, the retry pattern allows the calling service to automatically reattempt the operation a specified number of times. This can significantly improve reliability for intermittent failures.

Timeout

Setting appropriate timeouts for inter-service communication ensures that a slow or unresponsive service doesn't hold up other services indefinitely. If a response isn't received within the timeout period, the operation is aborted, preventing resource exhaustion and cascading delays.

Data Management Patterns for Microservices

Managing data in a distributed microservices environment presents unique challenges, especially concerning data consistency and transactional integrity. Each microservice typically owns its own data store, leading to a distributed data landscape.

Database per Service

This is a foundational pattern where each microservice has its own independent database. This allows services to choose the database technology best suited for their needs and ensures loose coupling, as services don't share database schemas. However, it complicates cross-service queries and transactional consistency.

Saga Pattern

For transactions that span multiple microservices, the Saga pattern provides a way to manage data consistency. A Saga is a sequence of local transactions where each transaction updates data within a single service. If a transaction fails, compensating transactions are executed to undo the preceding transactions, effectively rolling back the overall operation.

Event Sourcing

In Event Sourcing, all changes to application state are stored as a sequence of immutable events. The current state of the application can be reconstructed by replaying these events. This pattern is often used in conjunction with CQRS (Command Query Responsibility Segregation) and can be very powerful for auditing and debugging.

Deployment and Operational Patterns for Microservices

Deploying and managing microservices requires specialized strategies to handle their distributed nature, frequent updates, and operational complexity.

Service Discovery

In a dynamic microservices environment, services constantly change their network locations. Service discovery mechanisms (e.g., Eureka, Consul) allow services to register themselves and for clients to find their network addresses, enabling dynamic load balancing and fault tolerance.

Containerization and Orchestration

Technologies like Docker for containerization and Kubernetes for orchestration are almost synonymous with microservices deployment. Containers package services with their dependencies, ensuring consistency across environments. Orchestrators manage the deployment, scaling, and healing of these containers.

CI/CD Pipelines

Continuous Integration and Continuous Delivery (CI/CD) pipelines are essential for automating the

build, test, and deployment processes for each microservice independently. This allows for faster release cycles and reduces the risk of manual errors.

Leveraging GitHub for Microservices Patterns

GitHub has become an indispensable platform for the microservices community. It hosts a vast collection of open-source projects, libraries, frameworks, and, crucially, documentation related to microservices patterns. Developers and organizations often share their insights, implementations, and architectural blueprints on GitHub, making it a central repository for learning and adoption.

The platform's collaborative nature facilitates the refinement and evolution of these patterns. Code repositories often include detailed README files, wikis, and issue trackers that serve as excellent resources for understanding pattern implementations. The ability to fork, clone, and contribute to projects democratizes access to this valuable knowledge.

Finding Microservices Patterns PDFs on GitHub

Searching for "microservices patterns pdf github" on search engines will often lead you directly to repositories or Gists containing comprehensive PDF documents. Many experienced architects and companies publish detailed guides, cheat sheets, and pattern catalogs in PDF format and upload them to GitHub. These PDFs typically offer well-structured explanations, diagrams, and sometimes even code snippets illustrating the patterns.

Key strategies for finding these resources include:

- Using precise search terms like "microservices patterns cheat sheet pdf github" or "microservices design patterns book pdf github."

- Exploring popular repositories related to microservices or domain-driven design, as they often link to or contain such documents.
- Looking for Gists, which are often used for sharing single files like PDFs.
- Checking the "wiki" sections of prominent microservices-related projects on GitHub.

Practical Implementation with GitHub Examples

Beyond theoretical documentation in PDFs, GitHub is an excellent source for practical, real-world implementations of microservices patterns. Many open-source projects showcase how these patterns are applied in actual applications. By exploring the code, developers can learn:

- How a specific pattern is translated into code.
- The libraries and frameworks commonly used with certain patterns.
- Best practices for integrating different patterns within a microservices ecosystem.
- Architectural decisions and their trade-offs as demonstrated by the project structure.

Examining code from well-regarded microservices projects on GitHub can provide invaluable insights that static documentation might miss. It allows for hands-on learning and experimentation.

Common Challenges and Solutions in Microservices

Despite the benefits, microservices adoption comes with inherent challenges. Understanding these common pitfalls and their associated solutions is crucial for successful implementation.

- **Distributed Complexity:** Managing numerous independent services can be overwhelming. Solutions involve robust observability, well-defined APIs, and effective service discovery.
- **Data Consistency:** Maintaining data consistency across distributed databases is hard. Patterns like Sagas and Event Sourcing are key.
- **Testing:** Testing distributed systems is more complex than testing monoliths. Strategies include contract testing, integration testing, and end-to-end testing.
- **Operational Overhead:** Deploying and managing many services requires mature DevOps practices and robust automation. Containerization and orchestration are vital.
- **Inter-service Communication Failures:** Network latency and service unreliability are common. Resilience patterns like Circuit Breaker and Retry are essential.

Future Trends in Microservices Patterns

The field of microservices is continuously evolving. Emerging trends include the rise of serverless architectures, which can be seen as an extreme form of microservices, and the increasing adoption of service meshes (like Istio and Linkerd) to manage inter-service communication and observability in a platform-agnostic way. Machine learning is also playing a growing role in areas like automated anomaly detection and performance optimization within microservices. As the complexity of distributed

systems grows, so too will the need for more sophisticated and automated pattern implementations, often shared and developed within communities like those found on GitHub.

Frequently Asked Questions

What are the most common patterns discussed in popular microservices patterns PDFs or GitHub repositories?

Popular microservices patterns PDFs and GitHub repositories often cover core patterns like API Gateway, Service Discovery, Circuit Breaker, Database per Service, Event Sourcing, Saga, and CQRS. These are fundamental for building resilient and scalable microservice architectures.

Where can I find a comprehensive and up-to-date PDF or GitHub resource on microservices patterns?

Excellent resources can be found on platforms like GitHub (search for 'microservices patterns' or 'awesome microservices'), and official documentation from cloud providers like AWS, Azure, and Google Cloud. Look for repositories with high star counts and recent activity, and PDFs from reputable sources like Martin Fowler's site or established software engineering communities.

How are patterns like 'API Gateway' and 'Service Discovery' explained in typical microservices pattern documentation?

These resources usually explain the API Gateway as a single entry point for clients, abstracting the complexity of multiple microservices. Service Discovery is typically described as a mechanism for services to register themselves and for clients to find their network locations, often using solutions like Eureka, Consul, or Kubernetes' built-in service discovery.

What is the role of 'Database per Service' in microservices, and how is it typically illustrated in pattern guides?

The 'Database per Service' pattern, commonly found in PDFs and GitHub guides, emphasizes that each microservice should own its data exclusively, using its own database. This promotes loose coupling and independent development/deployment. Illustrations often show distinct database instances for each service.

Are there specific GitHub repositories that provide code examples for implementing microservices patterns?

Yes, many GitHub repositories offer practical code examples. Searching for terms like 'microservices patterns examples,' 'spring-boot microservices,' or specific pattern names alongside 'demo' or 'example' can yield repositories with working implementations in various languages and frameworks.

What advanced or less common microservices patterns might I find in more in-depth PDFs or GitHub resources?

More advanced resources might delve into patterns like Strangler Fig (for phased migration), Backend for Frontend (BFF), Branch by Abstraction, Anti-Corruption Layer, and various strategies for distributed tracing and observability, which are crucial for managing complex microservice ecosystems.

Additional Resources

Here are 9 book titles related to microservices patterns, with short descriptions, formatted as requested:

1. *Microservices Patterns: With examples in Java*
This foundational book delves deep into the most common and effective patterns for designing, developing, and deploying microservice architectures. It covers a wide range of topics, including

decomposing monolithic applications, handling inter-service communication, and managing distributed transactions. The extensive Java examples provide practical guidance for implementing these patterns in real-world scenarios.

2. *Building Microservices: Designing Fine-Grained Systems*
This practical guide offers a comprehensive look at the principles and practices behind building robust and scalable microservices. It emphasizes understanding the benefits of microservices while also acknowledging their inherent complexities. The book walks through the entire lifecycle of microservice development, from initial design considerations to deployment and ongoing maintenance.

3. *Microservices Architecture: An Essential Guide to Designing and Implementing Microservices*
This essential guide serves as a comprehensive resource for understanding the core concepts and architectural styles of microservices. It provides a structured approach to designing and implementing microservice-based systems, focusing on aspects like service decomposition, API design, and communication strategies. The book aims to equip developers and architects with the knowledge to build resilient and adaptable software systems.

4. *Microservices for Real-World Projects: Patterns and Best Practices*
This book focuses on the practical application of microservice patterns in real-world development scenarios. It bridges the gap between theoretical concepts and actual implementation, offering actionable advice and best practices. The content is geared towards helping teams navigate the challenges of adopting and managing microservices in production environments.

5. *Cloud Native Patterns: Designing and Discovering Microservices, Data, and Management*
While broader than just microservices, this book extensively covers patterns relevant to building cloud-native applications, where microservices are a central component. It explores patterns for service discovery, distributed tracing, resilience, and state management in distributed systems. The book is essential for anyone building modern applications that leverage cloud infrastructure.

6. *Microservices: From Design to Deployment*
This title suggests a holistic approach to microservices, covering the entire journey from initial design

decisions to the operational aspects of deployment. It likely explores various design patterns for breaking down systems into smaller, independent services. The book would also address the infrastructure and tooling necessary for successful deployment and management.

7. *Reactive Microservices: Building Distributed Systems with the Actor Model*
This book focuses on a specific paradigm for building microservices: the reactive approach, often implemented using the Actor Model. It details how to design highly available, fault-tolerant, and scalable microservices that can handle concurrency and distributed data effectively. The content is ideal for those interested in building responsive and resilient distributed systems.

8. *Microservices in .NET: Pattern-Based Approaches for Building Scalable Applications*
This title targets developers working within the .NET ecosystem, providing guidance on applying microservice patterns specifically within this framework. It would cover .NET-specific tools, libraries, and best practices for designing and implementing microservices. The book aims to help .NET developers leverage microservices for building scalable and maintainable applications.

9. *Hands-On Microservices with Docker and Kubernetes: Design, Build, and Deploy Scalable Microservice Applications*
This book takes a hands-on approach, integrating the practical aspects of microservices development with the essential tools of Docker and Kubernetes. It guides readers through the process of designing, building, and deploying microservice applications using these containerization and orchestration technologies. The focus is on practical implementation and leveraging these tools for efficient microservice management.

Microservices Patterns Pdf Github

Microservices Patterns: A Practical Guide (PDF & GitHub Repo)

Are you struggling to build scalable, maintainable, and resilient applications? Microservices architecture offers a powerful solution, but navigating its complexities can be daunting. Building, deploying, and managing a distributed system introduces a whole new set of challenges. Are you facing issues with inter-service communication, data consistency, or effective monitoring and logging in your microservices ecosystem? This comprehensive guide will equip you with the practical patterns and best practices needed to overcome these hurdles and build robust microservices-based applications.

This ebook, Microservices Patterns: A Practical Guide, provides a structured approach to mastering microservices architecture. It's complemented by a GitHub repository containing example code and configurations.

Contents:

Microservices Patterns: A Practical Guide - Article

Introduction: What are Microservices and Why Use Them?

Microservices architecture is an approach to software development where a large application is built as a collection of small, independent services. Each service focuses on a specific business function, is independently deployable, and can be written in different programming languages. This contrasts with monolithic architectures, where the entire application is a single, tightly coupled unit.

The move towards microservices is driven by several compelling reasons:

Improved Scalability: Individual services can be scaled independently based on their specific needs, optimizing resource utilization and cost-effectiveness. You can scale only the parts of your application experiencing high demand, avoiding unnecessary scaling of underutilized components.

Enhanced Agility and Faster Development Cycles: Smaller, well-defined services allow for faster development, testing, and deployment cycles. Teams can work independently on different services, promoting parallel development and faster releases.

Technology Diversity: Different services can be built using the most appropriate technology stack for their specific needs. This flexibility allows teams to choose the best tools for the job, improving development efficiency.

Fault Isolation: If one service fails, it doesn't necessarily bring down the entire application. This improves system resilience and availability.

Easier Maintenance and Updates: Smaller codebases are simpler to understand, maintain, and update. Changes to one service are less likely to have unintended consequences on other parts of the application.

However, microservices also present significant challenges:

Increased Complexity: Managing a distributed system is inherently more complex than managing a monolithic application. This complexity includes inter-service communication, data consistency, monitoring, and logging.

Operational Overhead: Deploying, monitoring, and managing many independent services requires robust infrastructure and tooling.

Data Consistency: Maintaining data consistency across multiple services can be challenging, requiring careful design and implementation of data management strategies.

Chapter 1: Design Patterns for Microservice Architecture

This chapter explores architectural styles and trade-offs relevant to microservice design. Key patterns include:

API Gateway: A central point of entry for all client requests, handling routing, authentication, and other cross-cutting concerns. It simplifies client interaction with the microservices landscape.

Backend for Frontend (BFF): Tailored backend services optimized for specific client types (e.g., mobile app, web application), providing customized responses and reducing data transfer overhead.

Event Sourcing: Storing data as a sequence of events, providing an immutable audit trail and enabling easier reconstruction of system state. This pattern facilitates asynchronous communication and simplifies data consistency management.

CQRS (Command Query Responsibility Segregation): Separating read and write operations, improving performance and scalability. Read operations can be optimized for speed, while write operations can focus on data consistency.

Saga Pattern: Managing distributed transactions across multiple microservices, ensuring data consistency even when individual services fail. This often uses event-driven communication to coordinate updates.

Chapter 2: Communication Strategies

Effective communication between microservices is crucial. This chapter covers:

Synchronous Communication (REST, gRPC): Suitable for real-time interactions where immediate responses are required. RESTful APIs are widely used, while gRPC offers higher performance for internal communication.

Asynchronous Communication (Message Queues, Event Buses): Ideal for decoupling services, improving resilience, and handling high volumes of messages. Message queues like RabbitMQ or Kafka, and event buses like Kafka Streams or Apache Pulsar, are commonly used.

Choosing the Right Communication Style: This section discusses factors influencing the choice of synchronous versus asynchronous communication, including performance requirements, coupling needs, and fault tolerance considerations.

Chapter 3: Data Management in Microservices

Data management in a distributed system presents unique challenges. This chapter covers:

Database per Service: Each microservice manages its own database, promoting autonomy and isolation.

Shared Databases (with caution): This approach simplifies data consistency but can create tight coupling and scalability limitations. It should be used judiciously.

Eventual Consistency: Accepting that data consistency may not be immediate, relying on asynchronous updates and conflict resolution mechanisms.

Data Replication and Synchronization: Techniques for maintaining data consistency across multiple databases, including master-slave replication and multi-master replication.

Chapter 4: Deployment and Orchestration

This chapter delves into the practicalities of deploying and managing microservices:

Containerization (Docker): Packaging services into containers for consistent execution across different environments.

Orchestration (Kubernetes): Automating the deployment, scaling, and management of containerized services.

CI/CD Pipelines: Automating the build, test, and deployment process for continuous integration and continuous delivery.

Service Discovery: Mechanisms for services to dynamically discover and connect to each other, essential in dynamic environments.

Chapter 5: Monitoring, Logging, and Tracing

Effective monitoring is critical for a distributed system. This chapter covers:

Centralized Logging: Aggregating logs from all services for efficient analysis and troubleshooting.

Distributed Tracing: Tracking requests as they flow through multiple services, identifying bottlenecks and performance issues.

Metrics and Dashboards: Collecting performance metrics (e.g., latency, throughput, error rates) and visualizing them using dashboards for real-time monitoring.

Chapter 6: Security in Microservices

Securing a microservices architecture requires a holistic approach. This chapter covers:

Authentication and Authorization: Securely identifying and verifying users and services.

Secure Communication (TLS/SSL): Protecting communication between services using encryption.

Data Security: Protecting data at rest and in transit.

Input Validation and Sanitization: Preventing security vulnerabilities by validating and sanitizing input data.

Chapter 7: Microservices Testing and Quality Assurance

Thorough testing is crucial for ensuring the quality and reliability of microservices. This chapter covers:

Unit Testing: Testing individual components within a service.

Integration Testing: Testing the interactions between multiple services.

End-to-End Testing: Testing the entire application flow, simulating real-world scenarios.

Contract Testing: Ensuring that services adhere to their defined contracts.

Conclusion: Future Trends and Best Practices Recap

This concluding chapter summarizes key takeaways, highlights emerging trends in microservices (serverless, service mesh), and reinforces best practices for successful microservices adoption.

FAQs:

1. What is the difference between microservices and monolithic architecture?
2. How do I choose the right communication style for my microservices?
3. What are the best practices for data management in a microservices environment?
4. What tools are available for monitoring and logging microservices?
5. How can I secure my microservices against attacks?
6. What are some common challenges in migrating to a microservices architecture?
7. What are the best practices for testing microservices?
8. How can I effectively manage the complexity of a microservices system?
9. What are the future trends in microservices architecture?

Related Articles:

1. Building a Microservices Architecture with Kubernetes: A practical guide to using Kubernetes for deploying and managing microservices.
2. Implementing an API Gateway for Microservices: A deep dive into designing and implementing an API gateway to simplify microservices access.
3. Microservices Communication Patterns: Synchronous vs. Asynchronous: Comparing different communication styles and their use cases.
4. Data Consistency Strategies for Microservices: Exploring techniques for maintaining data consistency across multiple services.
5. Monitoring and Logging Microservices with Prometheus and Grafana: A hands-on tutorial for monitoring microservices using these popular tools.
6. Securing Microservices with OAuth 2.0 and JWT: A comprehensive guide to secure authentication and authorization in microservices.
7. Testing Microservices: A Comprehensive Guide: Different approaches to testing at different levels of the microservices stack.
8. Best Practices for Deploying Microservices with CI/CD: A guide to setting up automated pipelines for continuous integration and continuous delivery.
9. Serverless Microservices: A Practical Approach: Exploring the use of serverless functions for building microservices.

microservices patterns pdf github: Microservices Patterns Chris Richardson, 2018-10-27 A comprehensive overview of the challenges teams face when moving to microservices, with industry-tested solutions to these problems. - Tim Moore, Lightbend 44 reusable patterns to develop and deploy reliable production-quality microservices-based applications, with worked examples in Java Key Features 44 design patterns for building and deploying microservices applications Drawing on decades of unique experience from author and microservice architecture pioneer Chris Richardson A pragmatic approach to the benefits and the drawbacks of microservices architecture Solve service decomposition, transaction management, and inter-service communication Purchase of the print book includes a free eBook in PDF, Kindle, and ePub formats from Manning Publications. About The Book Microservices Patterns teaches you 44 reusable patterns to reliably develop and deploy production-quality microservices-based applications. This invaluable set of design patterns builds on decades of distributed system experience, adding new patterns for composing services into systems that scale and perform under real-world conditions. More than just a patterns catalog, this practical guide with worked examples offers industry-tested advice to help you design, implement, test, and deploy your microservices-based application. What You Will Learn How (and why!) to use microservices architecture Service decomposition strategies Transaction management and querying patterns Effective testing strategies Deployment patterns This Book Is Written For Written for enterprise developers familiar with standard enterprise application architecture. Examples are in Java. About The Author Chris Richardson is a Java Champion, a JavaOne rock star, author of Manning’s POJOs in Action, and creator of the original CloudFoundry.com. Table of Contents Escaping monolithic hell Decomposition strategies Interprocess communication in a microservice architecture Managing transactions with sagas Designing business logic in a microservice architecture Developing business logic with event sourcing Implementing queries in a microservice architecture External API patterns Testing microservices: part 1 Testing microservices: part 2 Developing production-ready services Deploying microservices Refactoring to microservices

microservices patterns pdf github: Building Microservices with Go Nic Jackson, 2017-07-27 Your one-stop guide to the common patterns and practices, showing you how to apply these using the Go programming language About This Book This short, concise, and practical guide is packed with real-world examples of building microservices with Go It is easy to read and will benefit smaller teams who want to extend the functionality of their existing systems Using this practical approach will save your money in terms of maintaining a monolithic architecture and

demonstrate capabilities in ease of use Who This Book Is For You should have a working knowledge of programming in Go, including writing and compiling basic applications. However, no knowledge of RESTful architecture, microservices, or web services is expected. If you are looking to apply techniques to your own projects, taking your first steps into microservice architecture, this book is for you. What You Will Learn Plan a microservice architecture and design a microservice Write a microservice with a RESTful API and a database Understand the common idioms and common patterns in microservices architecture Leverage tools and automation that helps microservices become horizontally scalable Get a grounding in containerization with Docker and Docker-Compose, which will greatly accelerate your development lifecycle Manage and secure Microservices at scale with monitoring, logging, service discovery, and automation Test microservices and integrate API tests in Go In Detail Microservice architecture is sweeping the world as the de facto pattern to build web-based applications. Golang is a language particularly well suited to building them. Its strong community, encouragement of idiomatic style, and statically-linked binary artifacts make integrating it with other technologies and managing microservices at scale consistent and intuitive. This book will teach you the common patterns and practices, showing you how to apply these using the Go programming language. It will teach you the fundamental concepts of architectural design and RESTful communication, and show you patterns that provide manageable code that is supportable in development and at scale in production. We will provide you with examples on how to put these concepts and patterns into practice with Go. Whether you are planning a new application or working in an existing monolith, this book will explain and illustrate with practical examples how teams of all sizes can start solving problems with microservices. It will help you understand Docker and Docker-Compose and how it can be used to isolate microservice dependencies and build environments. We finish off by showing you various techniques to monitor, test, and secure your microservices. By the end, you will know the benefits of system resilience of a microservice and the advantages of Go stack. Style and approach The step-by-step tutorial focuses on building microservices. Each chapter expands upon the previous one, teaching you the main skills and techniques required to be a successful microservice practitioner.

microservices patterns pdf github: Production-Ready Microservices Susan J. Fowler, 2016-11-30 One of the biggest challenges for organizations that have adopted microservice architecture is the lack of architectural, operational, and organizational standardization. After splitting a monolithic application or building a microservice ecosystem from scratch, many engineers are left wondering what’s next. In this practical book, author Susan Fowler presents a set of microservice standards in depth, drawing from her experience standardizing over a thousand microservices at Uber. You’ll learn how to design microservices that are stable, reliable, scalable, fault tolerant, performant, monitored, documented, and prepared for any catastrophe. Explore production-readiness standards, including: Stability and Reliability: develop, deploy, introduce, and deprecate microservices; protect against dependency failures Scalability and Performance: learn essential components for achieving greater microservice efficiency Fault Tolerance and Catastrophe Preparedness: ensure availability by actively pushing microservices to fail in real time Monitoring: learn how to monitor, log, and display key metrics; establish alerting and on-call procedures Documentation and Understanding: mitigate tradeoffs that come with microservice adoption, including organizational sprawl and technical debt

microservices patterns pdf github: Design Patterns in PHP and Laravel Kelt Dockins, 2016-12-27 Learn each of the original gang of four design patterns, and how they are relevant to modern PHP and Laravel development. Written by a working developer who uses these patterns every day, you will easily be able to implement each pattern into your workflow and improve your development. Each pattern is covered with full examples of how it can be used. Too often design patterns are explained using tricky concepts, when in fact they are easy to use and can enrich your everyday development. Design Patterns in PHP and Laravel aims to break down tricky concepts into humorous and easy-to-recall details, so that you can begin using design patterns easily in your everyday work with PHP and Laravel. This book teaches you design patterns in PHP and Laravel

using real-world examples and plenty of humor. What You Will Learn Use the original gang of four design patterns in your PHP and Laravel development How each pattern should be used Solve problems when using the patterns Remember each pattern using mnemonics Who This Book Is For People using Laravel and PHP to do their job and want to improve their understanding of design patterns.

microservices patterns pdf github: *Embracing Microservices Design* Ovais Mehboob Ahmed Khan, Nabil Siddiqui, Timothy Oleson, Mark Fussell, 2021-10-29 Develop microservice-based enterprise applications with expert guidance to avoid failures and technological debt with the help of real-world examples Key FeaturesImplement the right microservices adoption strategy to transition from monoliths to microservicesExplore real-world use cases that explain anti-patterns and alternative practices in microservices developmentDiscover proven recommendations for avoiding architectural mistakes when designing microservicesBook Description Microservices have been widely adopted for designing distributed enterprise apps that are flexible, robust, and fine-grained into services that are independent of each other. There has been a paradigm shift where organizations are now either building new apps on microservices or transforming existing monolithic apps into microservices-based architecture. This book explores the importance of anti-patterns and the need to address flaws in them with alternative practices and patterns. You'll identify common mistakes caused by a lack of understanding when implementing microservices and cover topics such as organizational readiness to adopt microservices, domain-driven design, and resiliency and scalability of microservices. The book further demonstrates the anti-patterns involved in re-platforming brownfield apps and designing distributed data architecture. You'll also focus on how to avoid communication and deployment pitfalls and understand cross-cutting concerns such as logging, monitoring, and security. Finally, you'll explore testing pitfalls and establish a framework to address isolation, autonomy, and standardization. By the end of this book, you'll have understood critical mistakes to avoid while building microservices and the right practices to adopt early in the product life cycle to ensure the success of a microservices initiative. What you will learnDiscover the responsibilities of different individuals involved in a microservices initiativeAvoid the common mistakes in architecting microservices for scalability and resiliencyUnderstand the importance of domain-driven design when developing microservicesIdentify the common pitfalls involved in migrating monolithic applications to microservicesExplore communication strategies, along with their potential drawbacks and alternativesDiscover the importance of adopting governance, security, and monitoringUnderstand the role of CI/CD and testingWho this book is for This practical microservices book is for software architects, solution architects, and developers involved in designing microservices architecture and its development, who want to gain insights into avoiding pitfalls and drawbacks in distributed applications, and save time and money that might otherwise get wasted if microservices designs fail. Working knowledge of microservices is assumed to get the most out of this book.

microservices patterns pdf github: Hands-On Design Patterns with Kotlin Alexey Soshin, 2018-06-15 Make the most of Kotlin by leveraging design patterns and best practices to build scalable and high performing apps Key Features Understand traditional GOF design patterns to apply generic solutions Shift from OOP to FP; covering reactive and concurrent patterns in a step-by-step manner Choose the best microservices architecture and MVC for your development environment Book Description Design patterns enable you as a developer to speed up the development process by providing you with proven development paradigms. Reusing design patterns helps prevent complex issues that can cause major problems, improves your code base, promotes code reuse, and makes an architecture more robust. The mission of this book is to ease the adoption of design patterns in Kotlin and provide good practices for programmers. The book begins by showing you the practical aspects of smarter coding in Kotlin, explaining the basic Kotlin syntax and the impact of design patterns. From there, the book provides an in-depth explanation of the classical design patterns of creational, structural, and behavioral families, before heading into functional programming. It then takes you through reactive and concurrent patterns, teaching you about using

streams, threads, and coroutines to write better code along the way By the end of the book, you will be able to efficiently address common problems faced while developing applications and be comfortable working on scalable and maintainable projects of any size. What you will learn Get to grips with Kotlin principles, including its strengths and weaknesses Understand classical design patterns in Kotlin Explore functional programming using built-in features of Kotlin Solve real-world problems using reactive and concurrent design patterns Use threads and coroutines to simplify concurrent code flow Understand antipatterns to write clean Kotlin code, avoiding common pitfalls Learn about the design considerations necessary while choosing between architectures Who this book is for This book is for developers who would like to master design patterns with Kotlin to build efficient and scalable applications. Basic Java or Kotlin programming knowledge is assumed

microservices patterns pdf github: *Practical Microservices Architectural Patterns* Binildas Christudas, 2019-06-25 Take your distributed applications to the next level and see what the reference architectures associated with microservices can do for you. This book begins by showing you the distributed computing architecture landscape and provides an in-depth view of microservices architecture. Following this, you will work with CQRS, an essential pattern for microservices, and get a view of how distributed messaging works. Moving on, you will take a deep dive into Spring Boot and Spring Cloud. Coming back to CQRS, you will learn how event-driven microservices work with this pattern, using the Axon 2 framework. This takes you on to how transactions work with microservices followed by advanced architectures to address non-functional aspects such as high availability and scalability. In the concluding part of the book you develop your own enterprise-grade microservices application using the Axon framework and true BASE transactions, while making it as secure as possible. What You Will Learn Shift from monolith architecture to microservices Work with distributed and ACID transactionsBuild solid architectures without two-phase commit transactions Discover the high availability principles in microservices Who This Book Is For Java developers with basic knowledge of distributed and multi-threaded application architecture, and no knowledge of Spring Boot or Spring Cloud. Knowledge of CQRS and event-driven architecture is not mandatory as this book will cover these in depth.

microservices patterns pdf github: Building Microservices Sam Newman, 2015-02-02 Annotation Over the past 10 years, distributed systems have become more fine-grained. From the large multi-million line long monolithic applications, we are now seeing the benefits of smaller self-contained services. Rather than heavy-weight, hard to change Service Oriented Architectures, we are now seeing systems consisting of collaborating microservices. Easier to change, deploy, and if required retire, organizations which are in the right position to take advantage of them are yielding significant benefits. This book takes an holistic view of the things you need to be cognizant of in order to pull this off. It covers just enough understanding of technology, architecture, operations and organization to show you how to move towards finer-grained systems.

microservices patterns pdf github: *Microservice Patterns and Best Practices* Vinicius Feitosa Pacheco, 2018-01-31 Explore the concepts and tools you need to discover the world of microservices with various design patterns Key Features Get to grips with the microservice architecture and build enterprise-ready microservice applications Learn design patterns and the best practices while building a microservice application Obtain hands-on techniques and tools to create high-performing microservices resilient to possible fails Book Description Microservices are a hot trend in the development world right now. Many enterprises have adopted this approach to achieve agility and the continuous delivery of applications to gain a competitive advantage. This book will take you through different design patterns at different stages of the microservice application development along with their best practices. Microservice Patterns and Best Practices starts with the learning of microservices key concepts and showing how to make the right choices while designing microservices. You will then move onto internal microservices application patterns, such as caching strategy, asynchronism, CQRS and event sourcing, circuit breaker, and bulkheads. As you progress, you'll learn the design patterns of microservices. The book will guide you on where to use the perfect design pattern at the application development stage and how to break monolithic application

into microservices. You will also be taken through the best practices and patterns involved while testing, securing, and deploying your microservice application. At the end of the book, you will easily be able to create interoperable microservices, which are testable and prepared for optimum performance. What you will learn How to break monolithic application into microservices Implement caching strategies, CQRS and event sourcing, and circuit breaker patterns Incorporate different microservice design patterns, such as shared data, aggregator, proxy, and chained Utilize consolidate testing patterns such as integration, signature, and monkey tests Secure microservices with JWT, API gateway, and single sign on Deploy microservices with continuous integration or delivery, Blue-Green deployment Who this book is for This book is for architects and senior developers who would like implement microservice design patterns in their enterprise application development. The book assumes some prior programming knowledge.

microservices patterns pdf github: Learn Microservices with Spring Boot Moises Macero, 2017-12-08 Build a microservices architecture with Spring Boot, by evolving an application from a small monolith to an event-driven architecture composed of several services. This book follows an incremental approach to teach microservice structure, test-driven development, Eureka, Ribbon, Zuul, and end-to-end tests with Cucumber. Author Moises Macero follows a very pragmatic approach to explain the benefits of using this type of software architecture, instead of keeping you distracted with theoretical concepts. He covers some of the state-of-the-art techniques in computer programming, from a practical point of view. You'll focus on what's important, starting with the minimum viable product but keeping the flexibility to evolve it. What You'll Learn Build microservices with Spring Boot Use event-driven architecture and messaging with RabbitMQ Create RESTful services with Spring Master service discovery with Eureka and load balancing with Ribbon Route requests with Zuul as your API gateway Write end-to-end rests for an event-driven architecture using Cucumber Carry out continuous integration and deployment Who This Book Is For Those with at least some prior experience with Java programming. Some prior exposure to Spring Boot recommended but not required.

microservices patterns pdf github: Mastering Spring Boot 2.0 Dinesh Rajput, 2018-05-31 Learn to develop, test, and deploy your Spring Boot distributed application and explore various best practices. Key Features Build and deploy your microservices architecture in the cloud Build event-driven resilient systems using Hystrix and Turbine Explore API management tools such as KONG and API documentation tools such as Swagger Book Description Spring is one of the best frameworks on the market for developing web, enterprise, and cloud ready software. Spring Boot simplifies the building of complex software dramatically by reducing the amount of boilerplate code, and by providing production-ready features and a simple deployment model. This book will address the challenges related to power that come with Spring Boot's great configurability and flexibility. You will understand how Spring Boot configuration works under the hood, how to overwrite default configurations, and how to use advanced techniques to prepare Spring Boot applications to work in production. This book will also introduce readers to a relatively new topic in the Spring ecosystem – cloud native patterns, reactive programming, and applications. Get up to speed with microservices with Spring Boot and Spring Cloud. Each chapter aims to solve a specific problem or teach you a useful skillset. By the end of this book, you will be proficient in building and deploying your Spring Boot application. What you will learn Build logically structured and highly maintainable Spring Boot applications Configure RESTful microservices using Spring Boot Make the application production and operation-friendly with Spring Actuator Build modern, high-performance distributed applications using cloud patterns Manage and deploy your Spring Boot application to the cloud (AWS) Monitor distributed applications using log aggregation and ELK Who this book is for The book is targeted at experienced Spring and Java developers who have a basic knowledge of working with Spring Boot. The reader should be familiar with Spring Boot basics, and aware of its benefits over traditional Spring Framework-based applications.

microservices patterns pdf github: Kubernetes Patterns Bilgin Ibryam, Roland Huß, 2019-04-09 The way developers design, build, and run software has changed significantly with the

evolution of microservices and containers. These modern architectures use new primitives that require a different set of practices than most developers, tech leads, and architects are accustomed to. With this focused guide, Bilgin Ibryam and Roland Huß from Red Hat provide common reusable elements, patterns, principles, and practices for designing and implementing cloud-native applications on Kubernetes. Each pattern includes a description of the problem and a proposed solution with Kubernetes specifics. Many patterns are also backed by concrete code examples. This book is ideal for developers already familiar with basic Kubernetes concepts who want to learn common cloud native patterns. You'll learn about the following pattern categories: Foundational patterns cover the core principles and practices for building container-based cloud-native applications. Behavioral patterns explore finer-grained concepts for managing various types of container and platform interactions. Structural patterns help you organize containers within a pod, the atom of the Kubernetes platform. Configuration patterns provide insight into how application configurations can be handled in Kubernetes. Advanced patterns covers more advanced topics such as extending the platform with operators.

microservices patterns pdf github: Monolith to Microservices Sam Newman, 2019-11-14 How do you detangle a monolithic system and migrate it to a microservice architecture? How do you do it while maintaining business-as-usual? As a companion to Sam Newman's extremely popular Building Microservices, this new book details a proven method for transitioning an existing monolithic system to a microservice architecture. With many illustrative examples, insightful migration patterns, and a bevy of practical advice to transition your monolith enterprise into a microservice operation, this practical guide covers multiple scenarios and strategies for a successful migration, from initial planning all the way through application and database decomposition. You'll learn several tried and tested patterns and techniques that you can use as you migrate your existing architecture. Ideal for organizations looking to transition to microservices, rather than rebuild Helps companies determine whether to migrate, when to migrate, and where to begin Addresses communication, integration, and the migration of legacy systems Discusses multiple migration patterns and where they apply Provides database migration examples, along with synchronization strategies Explores application decomposition, including several architectural refactoring patterns Delves into details of database decomposition, including the impact of breaking referential and transactional integrity, new failure modes, and more

microservices patterns pdf github: Docker and Kubernetes for Java Developers Jaroslaw Krochmalski, 2017-08-30 Leverage the lethal combination of Docker and Kubernetes to automate deployment and management of Java applications About This Book Master using Docker and Kubernetes to build, deploy and manage Java applications in a jiff Learn how to create your own Docker image and customize your own cluster using Kubernetes Empower the journey from development to production using this practical guide. Who This Book Is For The book is aimed at Java developers who are eager to build, deploy, and manage applications very quickly using container technology. They need have no knowledge of Docker and Kubernetes. What You Will Learn Package Java applications into Docker images Understand the running of containers locally Explore development and deployment options with Docker Integrate Docker into Maven builds Manage and monitor Java applications running on Kubernetes clusters Create Continuous Delivery pipelines for Java applications deployed to Kubernetes In Detail Imagine creating and testing Java EE applications on Apache Tomcat Server or Wildfly Application server in minutes along with deploying and managing Java applications swiftly. Sounds too good to be true? But you have a reason to cheer as such scenarios are only possible by leveraging Docker and Kubernetes. This book will start by introducing Docker and delve deep into its networking and persistent storage concepts. You will then proceed to learn how to refactor monolith application into separate services by building an application and then packaging it into Docker containers. Next, you will create an image containing Java Enterprise Application and later run it using Docker. Moving on, the book will focus on Kubernetes and its features and you will learn to deploy a Java application to Kubernetes using Maven and monitor a Java application in production. By the end of the book, you will get hands-on

with some more advanced topics to further extend your knowledge about Docker and Kubernetes. Style and approach An easy-to-follow, practical guide that will help Java developers develop, deploy, and manage Java applications efficiently.

microservices patterns pdf github: *Microservices in Action* Morgan Bruce, Paulo A Pereira, 2018-10-03 The one [and only] book on implementing microservices with a real-world, cover-to-cover example you can relate to. - Christian Bach, Swiss Re Microservices in Action is a practical book about building and deploying microservice-based applications. Written for developers and architects with a solid grasp of service-oriented development, it tackles the challenge of putting microservices into production. Purchase of the print book includes a free eBook in PDF, Kindle, and ePub formats from Manning Publications. About the Technology Invest your time in designing great applications, improving infrastructure, and making the most out of your dev teams. Microservices are easier to write, scale, and maintain than traditional enterprise applications because they're built as a system of independent components. Master a few important new patterns and processes, and you'll be ready to develop, deploy, and run production-quality microservices. About the Book Microservices in Action teaches you how to write and maintain microservice-based applications. Created with day-to-day development in mind, this informative guide immerses you in real-world use cases from design to deployment. You'll discover how microservices enable an efficient continuous delivery pipeline, and explore examples using Kubernetes, Docker, and Google Container Engine. What's inside An overview of microservice architecture Building a delivery pipeline Best practices for designing multi-service transactions and queries Deploying with containers Monitoring your microservices About the Reader Written for intermediate developers familiar with enterprise architecture and cloud platforms like AWS and GCP. About the Author Morgan Bruce and Paulo A. Pereira are experienced engineering leaders. They work daily with microservices in a production environment, using the techniques detailed in this book. Table of Contents Designing and running microservices Microservices at SimpleBank Architecture of a microservice application Designing new features Transactions and queries in microservices Designing reliable services Building a reusable microservice framework Deploying microservices Deployment with containers and schedulers Building a delivery pipeline for microservices Building a monitoring system Using logs and traces to understand behavior Building microservice teams PART 1 - The lay of the land PART 2 - Design PART 3 - Deployment PART 4 - Observability and ownership

microservices patterns pdf github: *Microservices from Theory to Practice: Creating Applications in IBM Bluemix Using the Microservices Approach* Shahir Daya, Nguyen Van Duy, Kameswara Eati, Carlos M Ferreira, Dejan Glozic, Vasfi Gucer, Manav Gupta, Sunil Joshi, Valerie Lampkin, Marcelo Martins, Shishir Narain, Ramratan Vennam, IBM Redbooks, 2016-04-04 Microservices is an architectural style in which large, complex software applications are composed of one or more smaller services. Each of these microservices focuses on completing one task that represents a small business capability. These microservices can be developed in any programming language. They communicate with each other using language-neutral protocols, such as Representational State Transfer (REST), or messaging applications, such as IBM® MQ Light. This IBM Redbooks® publication gives a broad understanding of this increasingly popular architectural style, and provides some real-life examples of how you can develop applications using the microservices approach with IBM BluemixTM. The source code for all of these sample scenarios can be found on GitHub (https://github.com/). The book also presents some case studies from IBM products. We explain the architectural decisions made, our experiences, and lessons learned when redesigning these products using the microservices approach. Information technology (IT) professionals interested in learning about microservices and how to develop or redesign an application in Bluemix using microservices can benefit from this book.

microservices patterns pdf github: Design It! Michael Keeling, 2017-10-18 Don't engineer by coincidence-design it like you mean it! Filled with practical techniques, Design It! is the perfect introduction to software architecture for programmers who are ready to grow their design skills. Lead your team as a software architect, ask the right stakeholders the right questions, explore

design options, and help your team implement a system that promotes the right -ilities. Share your design decisions, facilitate collaborative design workshops that are fast, effective, and fun-and develop more awesome software! With dozens of design methods, examples, and practical know-how, Design It! shows you how to become a software architect. Walk through the core concepts every architect must know, discover how to apply them, and learn a variety of skills that will make you a better programmer, leader, and designer. Uncover the big ideas behind software architecture and gain confidence working on projects big and small. Plan, design, implement, and evaluate software architectures and collaborate with your team, stakeholders, and other architects. Identify the right stakeholders and understand their needs, dig for architecturally significant requirements, write amazing quality attribute scenarios, and make confident decisions. Choose technologies based on their architectural impact, facilitate architecture-centric design workshops, and evaluate architectures using lightweight, effective methods. Write lean architecture descriptions people love to read. Run an architecture design studio, implement the architecture you've designed, and grow your team's architectural knowledge. Good design requires good communication. Talk about your software architecture with stakeholders using whiteboards, documents, and code, and apply architecture-focused design methods in your day-to-day practice. Hands-on exercises, real-world scenarios, and practical team-based decision-making tools will get everyone on board and give you the experience you need to become a confident software architect.

microservices patterns pdf github: *gRPC: Up and Running* Kasun Indrasiri, Danesh Kuruppu, 2020-01-23 Get a comprehensive understanding of gRPC fundamentals through real-world examples. With this practical guide, you'll learn how this high-performance interprocess communication protocol is capable of connecting polyglot services in microservices architecture, while providing a rich framework for defining service contracts and data types. Complete with hands-on examples written in Go, Java, Node, and Python, this book also covers the essential techniques and best practices to use gRPC in production systems. Authors Kasun Indrasiri and Danesh Kuruppu discuss the importance of gRPC in the context of microservices development.

microservices patterns pdf github: Architecture Patterns with Python Harry Percival, Bob Gregory, 2020-03-05 As Python continues to grow in popularity, projects are becoming larger and more complex. Many Python developers are now taking an interest in high-level software design patterns such as hexagonal/clean architecture, event-driven architecture, and the strategic patterns prescribed by domain-driven design (DDD). But translating those patterns into Python isn't always straightforward. With this hands-on guide, Harry Percival and Bob Gregory from MADE.com introduce proven architectural design patterns to help Python developers manage application complexity—and get the most value out of their test suites. Each pattern is illustrated with concrete examples in beautiful, idiomatic Python, avoiding some of the verbosity of Java and C# syntax. Patterns include: Dependency inversion and its links to ports and adapters (hexagonal/clean architecture) Domain-driven design's distinction between entities, value objects, and aggregates Repository and Unit of Work patterns for persistent storage Events, commands, and the message bus Command-query responsibility segregation (CQRS) Event-driven architecture and reactive microservices

microservices patterns pdf github: Clean Architecture Robert C. Martin, 2017-09-12 Practical Software Architecture Solutions from the Legendary Robert C. Martin ("Uncle Bob") By applying universal rules of software architecture, you can dramatically improve developer productivity throughout the life of any software system. Now, building upon the success of his best-selling books Clean Code and The Clean Coder, legendary software craftsman Robert C. Martin ("Uncle Bob") reveals those rules and helps you apply them. Martin's Clean Architecture doesn't merely present options. Drawing on over a half-century of experience in software environments of every imaginable type, Martin tells you what choices to make and why they are critical to your success. As you've come to expect from Uncle Bob, this book is packed with direct, no-nonsense solutions for the real challenges you'll face–the ones that will make or break your projects. Learn what software architects need to achieve–and core disciplines and practices for achieving it Master essential software design

principles for addressing function, component separation, and data management See how programming paradigms impose discipline by restricting what developers can do Understand what's critically important and what's merely a "detail" Implement optimal, high-level structures for web, database, thick-client, console, and embedded applications Define appropriate boundaries and layers, and organize components and services See why designs and architectures go wrong, and how to prevent (or fix) these failures Clean Architecture is essential reading for every current or aspiring software architect, systems analyst, system designer, and software manager–and for every programmer who must execute someone else's designs. Register your product for convenient access to downloads, updates, and/or corrections as they become available.

microservices patterns pdf github: Learn Git in a Month of Lunches Rick Umali, 2015-09-01 Summary Learn Git in a Month of Lunches introduces the discipline of source code control using Git. Whether you're a newbie or a busy pro moving your source control to Git, you'll appreciate how this book concentrates on the components of Git you'll use every day. In easy-to-follow lessons designed to take an hour or less, you'll dig into Git's distributed collaboration model, along with core concepts like committing, branching, and merging. Purchase of the print book includes a free eBook in PDF, Kindle, and ePub formats from Manning Publications. About the Book Git is the source code control system preferred by modern development teams. Its decentralized architecture and lightning-fast branching let you concentrate on your code instead of tedious version control tasks. At first, Git may seem like a sprawling beast. Fortunately, to get started you just need to master a few essential techniques. Read on! Learn Git in a Month of Lunches introduces the discipline of source code control using Git. Helpful for both newbies who have never used source control and busy pros, this book concentrates on the components of Git you'll use every day. In easy-to-follow lessons that take an hour or less, you'll dig into Git's distributed collaboration model, along with core concepts like committing, branching, and merging. This book is a road map to the commands and processes you need to be instantly productive. What's Inside Start from square one—no experience required The most frequently used Git commands Mental models that show how Git works Learn when and how to branch code About the Reader No previous experience with Git or other source control systems is required. About the Author Rick Umali uses Git daily as a developer and is a skilled consultant, trainer, and speaker. Table of Contents Before you begin An overview of Git and version control Getting oriented with Git Making and using a Git repository Using Git with a GUI Tracking and updating files in Git Committing parts of changes The time machine that is Git Taking a fork in the road Merging branches Cloning Collaborating with remotes Pushing your changes Keeping in sync Software archaeology Understanding git rebase Workflows and branching conventions Working with GitHub Third-party tools and Git Sharpening your Git

microservices patterns pdf github: Jakarta EE Cookbook Elder Moraes, 2020-05-29 An enterprise Java developer's guide to learning JAX-RS, context and dependency injection, JavaServer Faces (JSF), and microservices with Eclipse MicroProfile using the latest features of Jakarta EE Key FeaturesExplore Jakarta EE's latest features and API specifications and discover their benefitsBuild and deploy microservices using Jakarta EE 8 and Eclipse MicroProfileBuild robust RESTful web services for various enterprise scenarios using the JAX-RS, JSON-P, and JSON-B APIsBook Description Jakarta EE is widely used around the world for developing enterprise applications for a variety of domains. With this book, Java professionals will be able to enhance their skills to deliver powerful enterprise solutions using practical recipes. This second edition of the Jakarta EE Cookbook takes you through the improvements introduced in its latest version and helps you get hands-on with its significant APIs and features used for server-side development. You'll use Jakarta EE for creating RESTful web services and web applications with the JAX-RS, JSON-P, and JSON-B APIs and learn how you can improve the security of your enterprise solutions. Not only will you learn how to use the most important servers on the market, but you'll also learn to make the best of what they have to offer for your project. From an architectural point of view, this Jakarta book covers microservices, cloud computing, and containers. It allows you to explore all the tools for building

reactive applications using Jakarta EE and core Java features such as lambdas. Finally, you'll discover how professionals can improve their projects by engaging with and contributing to the community. By the end of this book, you'll have become proficient in developing and deploying enterprise applications using Jakarta EE. What you will learnWork with Jakarta EE's most commonly used APIs and features for server-side developmentEnable fast and secure communication in web applications with the help of HTTP2Build enterprise applications with reusable componentsBreak down monoliths into microservices using Jakarta EE and Eclipse MicroProfileImprove your enterprise applications with multithreading and concurrencyRun applications in the cloud with the help of containersGet to grips with continuous delivery and deployment for shipping your applications effectivelyWho this book is for This book is for Java EE developers who want to build enterprise applications or update their legacy apps with Jakarta EE's latest features and specifications. Some experience of working with Java EE and knowledge of web and cloud computing will assist with understanding the concepts covered in this book.

microservices patterns pdf github: Enterprise Java Microservices Kenneth Finnigan, 2018-09-27 Summary Enterprise Java Microservices is an example-rich tutorial that shows how to design and manage large-scale Java applications as a collection of microservices. Purchase of the print book includes a free eBook in PDF, Kindle, and ePub formats from Manning Publications. About the Technology Large applications are easier to develop and maintain when you build them from small, simple components. Java developers now enjoy a wide range of tools that support microservices application development, including right-sized app servers, open source frameworks, and well-defined patterns. Best of all, you can build microservices applications using your existing Java skills. About the Book Enterprise Java Microservices teaches you to design and build JVM-based microservices applications. You'll start by learning how microservices designs compare to traditional Java EE applications. Always practical, author Ken Finnigan introduces big-picture concepts along with the tools and techniques you'll need to implement them. You'll discover ecosystem components like Netflix Hystrix for fault tolerance and master the Just enough Application Server (JeAS) approach. To ensure smooth operations, you'll also examine monitoring, security, testing, and deploying to the cloud. What's inside The microservices mental model Cloud-native development Strategies for fault tolerance and monitoring Securing your finished applications About the Reader This book is for Java developers familiar with Java EE. About the Author Ken Finnigan leads the Thorntail project at Red Hat, which seeks to make developing microservices for the cloud with Java and Java EE as easy as possible. Table of Contents PART 1 MICROSERVICES BASICS Enterprise Java microservices Developing a simple RESTful microservice Just enough Application Server for microservices Microservices testing Cloud native development PART 2 - IMPLEMENTING ENTERPRISE JAVA MICROSERVICES Consuming microservices Discovering microservices for consumption Strategies for fault tolerance and monitoring Securing a microservice Architecting a microservice hybrid Data streaming with Apache Kafka

microservices patterns pdf github: Microservices: Up and Running Ronnie Mitra, Irakli Nadareishvili, 2020-11-25 Microservices architectures offer faster change speeds, better scalability, and cleaner, evolvable system designs. But implementing your first microservices architecture is difficult. How do you make myriad choices, educate your team on all the technical details, and navigate the organization to a successful execution to maximize your chance of success? With this book, authors Ronnie Mitra and Irakli Nadareishvili provide step-by-step guidance for building an effective microservices architecture. Architects and engineers will follow an implementation journey based on techniques and architectures that have proven to work for microservices systems. You'll build an operating model, a microservices design, an infrastructure foundation, and two working microservices, then put those pieces together as a single implementation. For anyone tasked with building microservices or a microservices architecture, this guide is invaluable. Learn an effective and explicit end-to-end microservices system design Define teams, their responsibilities, and guidelines for working together Understand how to slice a big application into a collection of microservices Examine how to isolate and embed data into corresponding microservices Build a

simple yet powerful CI/CD pipeline for infrastructure changes Write code for sample microservices Deploy a working microservices application on Amazon Web Services

microservices patterns pdf github: Microservices Eberhard Wolff, 2016-10-03 The Most Complete, Practical, and Actionable Guide to Microservices Going beyond mere theory and marketing hype, Eberhard Wolff presents all the knowledge you need to capture the full benefits of this emerging paradigm. He illuminates microservice concepts, architectures, and scenarios from a technology-neutral standpoint, and demonstrates how to implement them with today's leading technologies such as Docker, Java, Spring Boot, the Netflix stack, and Spring Cloud. The author fully explains the benefits and tradeoffs associated with microservices, and guides you through the entire project lifecycle: development, testing, deployment, operations, and more. You'll find best practices for architecting microservice-based systems, individual microservices, and nanoservices, each illuminated with pragmatic examples. The author supplements opinions based on his experience with concise essays from other experts, enriching your understanding and illuminating areas where experts disagree. Readers are challenged to experiment on their own the concepts explained in the book to gain hands-on experience. Discover what microservices are, and how they differ from other forms of modularization Modernize legacy applications and efficiently build new systems Drive more value from continuous delivery with microservices Learn how microservices differ from SOA Optimize the microservices project lifecycle Plan, visualize, manage, and evolve architecture Integrate and communicate among microservices Apply advanced architectural techniques, including CQRS and Event Sourcing Maximize resilience and stability Operate and monitor microservices in production Build a full implementation with Docker, Java, Spring Boot, the Netflix stack, and Spring Cloud Explore nanoservices with Amazon Lambda, OSGi, Java EE, Vert.x, Erlang, and Seneca Understand microservices' impact on teams, technical leaders, product owners, and stakeholders Managers will discover better ways to support microservices, and learn how adopting the method affects the entire organization. Developers will master the technical skills and concepts they need to be effective. Architects will gain a deep understanding of key issues in creating or migrating toward microservices, and exactly what it will take to transform their plans into reality.

microservices patterns pdf github: Git in Practice Mike McQuaid, 2014-09-29 Summary Git in Practice is a collection of 66 tested techniques that will optimize the way you and your team manage your development projects. The book begins with a brief reminder of the core version control concepts you need when using Git and moves on to the high-value features you may not have explored yet. Then, you'll dig into cookbook-style techniques like history visualization, advanced branching and rewriting history each presented in a problem-solution-discussion format. Finally you'll work out how to use Git to its full potential through configuration, team workflows, submodules and using GitHub pull requests effectively. Purchase of the print book includes a free eBook in PDF, Kindle, and ePub formats from Manning Publications. About the Technology Git is a source control system, but it's a lot more than just that. For teams working in today's agile, continuous delivery environments, Git is a strategic advantage. Built with a decentralized structure that's perfect for a distributed team, Git manages branching, committing, complex merges, and task switching with minimal ceremony so you can concentrate on your code. About the Book Git in Practice is a collection of battle-tested techniques designed to optimize the way you and your team manage development projects. After a brief overview of Git's core features, this practical guide moves quickly to high-value topics like history visualization, advanced branching and rewriting, optimized configuration, team workflows, submodules, and how to use GitHub pull requests. Written in an easy-to-follow Problem/Solution/Discussion format with numerous diagrams and examples, it skips the theory and gets right to the nitty-gritty tasks that will transform the way you work. Written for developers familiar with version control and ready for the good stuff in Git. What's Inside Team interaction strategies and techniques Replacing bad habits with good practices Juggling complex configurations Rewriting history and disaster recovery About the Author Mike McQuaid is a software engineer at GitHub. He's contributed to Qt and the Linux kernel, and he maintains the Git-based Homebrew project. Table of Contents PART 1 INTRODUCTION TO GIT Local Git Remote Git PART 2

GIT ESSENTIALS Filesystem interactions History visualization Advanced branching Rewriting history and disaster recovery PART 3 ADVANCED GIT Personalizing Git Vendoring dependencies as submodules Working with Subversion GitHub pull requests Hosting a repository PART 4 GIT BEST PRACTICES Creating a clean history Merging vs. rebasing Recommended team workflows

microservices patterns pdf github: Data Management at Scale Piethein Strengholt, 2020-07-29 As data management and integration continue to evolve rapidly, storing all your data in one place, such as a data warehouse, is no longer scalable. In the very near future, data will need to be distributed and available for several technological solutions. With this practical book, you'll learnhow to migrate your enterprise from a complex and tightly coupled data landscape to a more flexible architecture ready for the modern world of data consumption. Executives, data architects, analytics teams, and compliance and governance staff will learn how to build a modern scalable data landscape using the Scaled Architecture, which you can introduce incrementally without a large upfront investment. Author Piethein Strengholt provides blueprints, principles, observations, best practices, and patterns to get you up to speed. Examine data management trends, including technological developments, regulatory requirements, and privacy concerns Go deep into the Scaled Architecture and learn how the pieces fit together Explore data governance and data security, master data management, self-service data marketplaces, and the importance of metadata

microservices patterns pdf github: *Microservices and Containers* Parminder Singh Kocher, 2018-03-16 Transition to Microservices and DevOps to Transform Your Software Development Effectiveness Thanks to the tech sector's latest game-changing innovations—the Internet of Things (IoT), software-enabled networking, and software as a service (SaaS), to name a few—there is now a seemingly insatiable demand for platforms and architectures that can improve the process of application development and deployment. In Microservices and Containers, longtime systems architect and engineering team leader Parminder Kocher analyzes two of the hottest new technology trends: microservices and containers. Together, as Kocher demonstrates, microservices and Docker containers can bring unprecedented agility and scalability to application development and deployment, especially in large, complex projects where speed is crucial but small errors can be disastrous. Learn how to leverage microservices and Docker to drive modular architectural design, on-demand scalability, application performance and reliability, time-to-market, code reuse, and exponential improvements in DevOps effectiveness. Kocher offers detailed guidance and a complete roadmap for transitioning from monolithic architectures, as well as an in-depth case study that walks the reader through the migration of an enterprise-class SOA system. Understand how microservices enable you to organize applications into standalone components that are easier to manage, update, and scale Decide whether microservices and containers are worth your investment, and manage the organizational learning curve associated with them Apply best practices for interprocess communication among microservices Migrate monolithic systems in an orderly fashion Understand Docker containers, installation, and interfaces Network, orchestrate, and manage Docker containers effectively Use Docker to maximize scalability in microservices-based applications Apply your learning with an in-depth, hands-on case study Whether you are a software architect/developer or systems professional looking to move on from older approaches or a manager trying to maximize the business value of these technologies, Microservices and Containers will be an invaluable addition to your library. Register your product at informit.com/register for convenient access to downloads, updates, and/or corrections as they become available.

microservices patterns pdf github: Building Microservices with .NET Core Gaurav Kumar Aroraa, Lalit Kale, Kanwar Manish, 2017-06-14 Architect your .NET applications by breaking them into really small pieces—microservices—using this practical, example-based guide About This Book Start your microservices journey and understand a broader perspective of microservices development Build, deploy, and test microservices using ASP.Net MVC, Web API, and Microsoft Azure Cloud Get started with reactive microservices and understand the fundamentals behind it Who This Book Is For This book is for .NET Core developers who want to learn and understand microservices architecture and implement it in their .NET Core applications. It's ideal for developers

who are completely new to microservices or have just a theoretical understanding of this architectural approach and want to gain a practical perspective in order to better manage application complexity. What You Will Learn Compare microservices with monolithic applications and SOA Identify the appropriate service boundaries by mapping them to the relevant bounded contexts Define the service interface and implement the APIs using ASP.NET Web API Integrate the services via synchronous and asynchronous mechanisms Implement microservices security using Azure Active Directory, OpenID Connect, and OAuth 2.0 Understand the operations and scaling of microservices in .NET Core Understand the testing pyramid and implement consumer-driven contract using pact net core Understand what the key features of reactive microservices are and implement them using reactive extension In Detail Microservices is an architectural style that promotes the development of complex applications as a suite of small services based on business capabilities. This book will help you identify the appropriate service boundaries within the business. We'll start by looking at what microservices are, and what the main characteristics are. Moving forward, you will be introduced to real-life application scenarios, and after assessing the current issues, we will begin the journey of transforming this application by splitting it into a suite of microservices. You will identify the service boundaries, split the application into multiple microservices, and define the service contracts. You will find out how to configure, deploy, and monitor microservices, and configure scaling to allow the application to quickly adapt to increased demand in the future. With an introduction to the reactive microservices, you strategically gain further value to keep your code base simple, focusing on what is more important rather than the messy asynchronous calls. Style and approach This guide serves as a stepping stone that helps .NET Core developers in their microservices architecture. This book provides just enough theory to understand the concepts and apply the examples.

microservices patterns pdf github: Microservices with Clojure Anuj Kumar, 2018-01-25 The common patterns and practices of the microservice architecture and their application using the Clojure programming language. Key Features Relevance of the microservice architecture and benefits of Clojure's functional and simple features to implement it. Learn best practices and common principles to avoid common pitfalls while developing microservices. Learn how to use Pedestal to build your next microservices, secure them using JWT, and monitor them using the ELK stack Book Description The microservice architecture is sweeping the world as the de facto pattern with which to design and build scalable, easy-tomaintain web applications. This book will teach you common patterns and practices, and will show you how to apply these using the Clojure programming language. This book will teach you the fundamental concepts of architectural design and RESTful communication, and show you patterns that provide manageable code that is supportable in development and at scale in production. We will provide you with examples of how to put these concepts and patterns into practice with Clojure. This book will explain and illustrate, with practical examples, how teams of all sizes can start solving problems with microservices. You will learn the importance of writing code that is asynchronous and non-blocking and how Pedestal helps us do this. Later, the book explains how to build Reactive microservices in Clojure that adhere to the principles underlying the Reactive Manifesto. We finish off by showing you various ways to monitor, test, and secure your microservices. By the end, you will be fully capable of setting up, modifying, and deploying a microservice with Clojure and Pedestal. What you will learn Explore the pros and cons of monolithic and microservice architectures Use Clojure to effectively build a reallife application using Microservices Gain practical knowledge of the Clojure Pedestal framework and how to use it to build Microservices Explore various persistence patterns and learn how to use Apache Kafka to build event-driven microservice architectures Secure your Microservices using JWT Monitor Microservices at scale using the ELK stack Deploy Microservices at scale using container orchestration platforms such as Kubernetes Who this book is for You should have a working knowledge of programming in Clojure. However, no knowledge of RESTful architecture, microservices, or web services is expected. If you are looking to apply techniques to your own projects, taking your first steps into microservice architecture, this book is for you.

microservices patterns pdf github: Software Architecture with C++ Adrian Ostrowski, Piotr Gaczkowski, 2021-04-23 Apply business requirements to IT infrastructure and deliver a high-quality product by understanding architectures such as microservices, DevOps, and cloud-native using modern C++ standards and features Key FeaturesDesign scalable large-scale applications with the C++ programming languageArchitect software solutions in a cloud-based environment with continuous integration and continuous delivery (CI/CD)Achieve architectural goals by leveraging design patterns, language features, and useful toolsBook Description Software architecture refers to the high-level design of complex applications. It is evolving just like the languages we use, but there are architectural concepts and patterns that you can learn to write high-performance apps in a high-level language without sacrificing readability and maintainability. If you're working with modern C++, this practical guide will help you put your knowledge to work and design distributed, large-scale apps. You'll start by getting up to speed with architectural concepts, including established patterns and rising trends, then move on to understanding what software architecture actually is and start exploring its components. Next, you'll discover the design concepts involved in application architecture and the patterns in software development, before going on to learn how to build, package, integrate, and deploy your components. In the concluding chapters, you'll explore different architectural qualities, such as maintainability, reusability, testability, performance, scalability, and security. Finally, you will get an overview of distributed systems, such as service-oriented architecture, microservices, and cloud-native, and understand how to apply them in application development. By the end of this book, you'll be able to build distributed services using modern C++ and associated tools to deliver solutions as per your clients' requirements. What you will learnUnderstand how to apply the principles of software architectureApply design patterns and best practices to meet your architectural goalsWrite elegant, safe, and performant code using the latest C++ featuresBuild applications that are easy to maintain and deployExplore the different architectural approaches and learn to apply them as per your requirementSimplify development and operations using application containersDiscover various techniques to solve common problems in software design and developmentWho this book is for This software architecture C++ programming book is for experienced C++ developers looking to become software architects or develop enterprise-grade applications.

microservices patterns pdf github: Pro C# 5.0 and the .NET 4.5 Framework Andrew Troelsen, 2012-10-07 This new edition of Pro C# 5.0 and the .NET 4.5 Platform has been completely revised and rewritten to reflect the latest changes to the C# language specification and new advances in the .NET Framework. You'll find new chapters covering all the important new features that make .NET 4.5 the most comprehensive release yet, including: .NET APIs for Windows 8 style UI apps New asynchronous task-based model for async operations How HTML5 support is being wrapped into C# web applications New programming interfaces for HTTP applications, including improved IPv6 support Expanded WPF, WCF and WF libraries giving C# more power than ever before This comes on top of award winning coverage of core C# features, both old and new, that have made the previous editions of this book so popular (you'll find everything from generics to pLINQ covered here). The mission of this text is to provide you with a rock-solid foundation in the C# programming language and the core aspects of the .NET platform (assemblies, remoting, Windows Forms, Web Forms, ADO.NET, XML web services, etc.). Once you digest the information presented in these 25 chapters, you'll be in a perfect position to apply this knowledge to your specific programming assignments, and you'll be well equipped to explore the .NET universe on your own terms.

microservices patterns pdf github: Data Pipelines Pocket Reference James Densmore, 2021-02-10 Data pipelines are the foundation for success in data analytics. Moving data from numerous diverse sources and transforming it to provide context is the difference between having data and actually gaining value from it. This pocket reference defines data pipelines and explains how they work in today's modern data stack. You'll learn common considerations and key decision points when implementing pipelines, such as batch versus streaming data ingestion and build versus

buy. This book addresses the most common decisions made by data professionals and discusses foundational concepts that apply to open source frameworks, commercial products, and homegrown solutions. You'll learn: What a data pipeline is and how it works How data is moved and processed on modern data infrastructure, including cloud platforms Common tools and products used by data engineers to build pipelines How pipelines support analytics and reporting needs Considerations for pipeline maintenance, testing, and alerting

microservices patterns pdf github: *Microservices Security in Action* Wajjakkara Kankanamge Anthony Nuwan Dias, Prabath Siriwardena, 2020-07-11 ”A complete guide to the challenges and solutions in securing microservices architectures.” —Massimo Siani, FinDynamic Key Features Secure microservices infrastructure and code Monitoring, access control, and microservice-to-microservice communications Deploy securely using Kubernetes, Docker, and the Istio service mesh. Hands-on examples and exercises using Java and Spring Boot Purchase of the print book includes a free eBook in PDF, Kindle, and ePub formats from Manning Publications. Microservices Security in Action teaches you how to address microservices-specific security challenges throughout the system. This practical guide includes plentiful hands-on exercises using industry-leading open-source tools and examples using Java and Spring Boot. About The Book Design and implement security into your microservices from the start. Microservices Security in Action teaches you to assess and address security challenges at every level of a Microservices application, from APIs to infrastructure. You’ll find effective solutions to common security problems, including throttling and monitoring, access control at the API gateway, and microservice-to-microservice communication. Detailed Java code samples, exercises, and real-world business use cases ensure you can put what you’ve learned into action immediately. What You Will Learn Microservice security concepts Edge services with an API gateway Deployments with Docker, Kubernetes, and Istio Security testing at the code level Communications with HTTP, gRPC, and Kafka This Book Is Written For For experienced microservices developers with intermediate Java skills. About The Author Prabath Siriwardena is the vice president of security architecture at WSO2. Nuwan Dias is the director of API architecture at WSO2. They have designed secure systems for many Fortune 500 companies. Table of Contents PART 1 OVERVIEW 1 Microservices security landscape 2 First steps in securing microservices PART 2 EDGE SECURITY 3 Securing north/south traffic with an API gateway 4 Accessing a secured microservice via a single-page application 5 Engaging throttling, monitoring, and access control PART 3 SERVICE-TO-SERVICE COMMUNICATIONS 6 Securing east/west traffic with certificates 7 Securing east/west traffic with JWT 8 Securing east/west traffic over gRPC 9 Securing reactive microservices PART 4 SECURE DEPLOYMENT 10 Conquering container security with Docker 11 Securing microservices on Kubernetes 12 Securing microservices with Istio service mesh PART 5 SECURE DEVELOPMENT 13 Secure coding practices and automation

microservices patterns pdf github: Domain-driven Design Eric Evans, 2004 Domain-Driven Design incorporates numerous examples in Java-case studies taken from actual projects that illustrate the application of domain-driven design to real-world software development.

microservices patterns pdf github: Building Event-Driven Microservices Adam Bellemare, 2020-07-02 Organizations today often struggle to balance business requirements with ever-increasing volumes of data. Additionally, the demand for leveraging large-scale, real-time data is growing rapidly among the most competitive digital industries. Conventional system architectures may not be up to the task. With this practical guide, you’ll learn how to leverage large-scale data usage across the business units in your organization using the principles of event-driven microservices. Author Adam Bellemare takes you through the process of building an event-driven microservice-powered organization. You’ll reconsider how data is produced, accessed, and propagated across your organization. Learn powerful yet simple patterns for unlocking the value of this data. Incorporate event-driven design and architectural principles into your own systems. And completely rethink how your organization delivers value by unlocking near-real-time access to data at scale. You’ll learn: How to leverage event-driven architectures to deliver exceptional business value The role of microservices in supporting event-driven designs Architectural patterns to ensure

success both within and between teams in your organization Application patterns for developing powerful event-driven microservices Components and tooling required to get your microservice ecosystem off the ground

microservices patterns pdf github: *Java 9 Modularity* Sander Mak, Paul Bakker, 2017-09-07 The upcoming Java 9 module system will affect existing applications and offer new ways of creating modular and maintainable applications. With this hands-on book, Java developers will learn not only about the joys of modularity, but also about the patterns needed to create truly modular and reliable applications. Authors Sander Mak and Paul Bakker teach you the concepts behind the Java 9 module system, along with the new tools it offers. You'll also learn how to modularize existing code and how to build new Java applications in a modular way. Understand Java 9 module system concepts Master the patterns and practices for building truly modular applications Migrate existing applications and libraries to Java 9 modules Use JDK 9 tools for modular development and migration

microservices patterns pdf github: *Bootstrapping Microservices with Docker, Kubernetes, and Terraform* Ashley Davis, 2021-03-09 Summary The best way to learn microservices development is to build something! Bootstrapping Microservices with Docker, Kubernetes, and Terraform guides you from zero through to a complete microservices project, including fast prototyping, development, and deployment. You'll get your feet wet using industry-standard tools as you learn and practice the practical skills you'll use for every microservices application. Following a true bootstrapping approach, you'll begin with a simple, familiar application and build up your knowledge and skills as you create and deploy a real microservices project. Purchase of the print book includes a free eBook in PDF, Kindle, and ePub formats from Manning Publications. About the technology Taking microservices from proof of concept to production is a complex, multi-step operation relying on tools like Docker, Terraform, and Kubernetes for packaging and deployment. The best way to learn the process is to build a project from the ground up, and that's exactly what you'll do with this book! About the book In Bootstrapping Microservices with Docker, Kubernetes, and Terraform, author Ashley Davis lays out a comprehensive approach to building microservices. You'll start with a simple design and work layer-by-layer until you've created your own video streaming application. As you go, you'll learn to configure cloud infrastructure with Terraform, package microservices using Docker, and deploy your finished project to a Kubernetes cluster. What's inside Developing and testing microservices applications Working with cloud providers Applying automated testing Implementing infrastructure as code and setting up a continuous delivery pipeline Monitoring, managing, and troubleshooting About the reader Examples are in JavaScript. No experience with microservices, Kubernetes, Terraform, or Docker required. About the author Ashley Davis is a software developer, entrepreneur, stock trader, and the author of Manning's Data Wrangling with JavaScript. Table of Contents 1 Why microservices? 2 Creating your first microservice 3 Publishing your first microservice 4 Data management for microservices 5 Communication between microservices 6 Creating your production environment 7 Getting to continuous delivery 8 Automated testing for microservices 9 Exploring FlixTube 10 Healthy microservices 11 Pathways to scalability

microservices patterns pdf github: Design Patterns for Cloud Native Applications Kasun Indrasiri, Sriskandarajah Suhothayan, 2021-05-17 With the immense cost savings and scalability the cloud provides, the rationale for building cloud native applications is no longer in question. The real issue is how. With this practical guide, developers will learn about the most commonly used design patterns for building cloud native applications using APIs, data, events, and streams in both greenfield and brownfield development. You'll learn how to incrementally design, develop, and deploy large and effective cloud native applications that you can manage and maintain at scale with minimal cost, time, and effort. Authors Kasun Indrasiri and Sriskandarajah Suhothayan highlight use cases that effectively demonstrate the challenges you might encounter at each step. Learn the fundamentals of cloud native applications Explore key cloud native communication, connectivity, and composition patterns Learn decentralized data management techniques Use event-driven architecture to build distributed and scalable cloud native applications Explore the most commonly used patterns for API management and consumption Examine some of the tools and technologies

you'll need for building cloud native systems

microservices patterns pdf github: *Cloud Native Patterns* Cornelia Davis, 2019-05-12 Summary Cloud Native Patternsis your guide to developing strong applications that thrive in the dynamic, distributed, virtual world of the cloud. This book presents a mental model for cloud-native applications, along with the patterns, practices, and tooling that set them apart. Purchase of the print book includes a free eBook in PDF, Kindle, and ePub formats from Manning Publications. About the Technology Cloud platforms promise the holy grail: near-zero downtime, infinite scalability, short feedback cycles, fault-tolerance, and cost control. But how do you get there? By applying cloudnative designs, developers can build resilient, easily adaptable, web-scale distributed applications that handle massive user traffic and data loads. Learn these fundamental patterns and practices, and you'll be ready to thrive in the dynamic, distributed, virtual world of the cloud. About the Book With 25 years of experience under her belt, Cornelia Davis teaches you the practices and patterns that set cloud-native applications apart. With realistic examples and expert advice for working with apps, data, services, routing, and more, she shows you how to design and build software that functions beautifully on modern cloud platforms. As you read, you will start to appreciate that cloud-native computing is more about the how and why rather than the where. What's inside The lifecycle of cloud-native apps Cloud-scale configuration management Zero downtime upgrades, versioned services, and parallel deploys Service discovery and dynamic routing Managing interactions between services, including retries and circuit breakers About the Reader Requires basic software design skills and an ability to read Java or a similar language. About the Author Cornelia Davis is Vice President of Technology at Pivotal Software. A teacher at heart, she's spent the last 25 years making good software and great software developers. Table of Contents PART 1 - THE CLOUD-NATIVE CONTEXT You keep using that word: Defining cloud-native Running cloud-native applications in production The platform for cloud-native software PART 2 - CLOUD-NATIVE PATTERNS Event-driven microservices: It's not just request/response App redundancy: Scale-out and statelessness Application configuration: Not just environment variables The application lifecycle: Accounting for constant change Accessing apps: Services, routing, and service discovery Interaction redundancy: Retries and other control loops Fronting services: Circuit breakers and API gateways Troubleshooting: Finding the needle in the haystack Cloud-native data: Breaking the data monolith

Back to Home: https://new.teachat.com