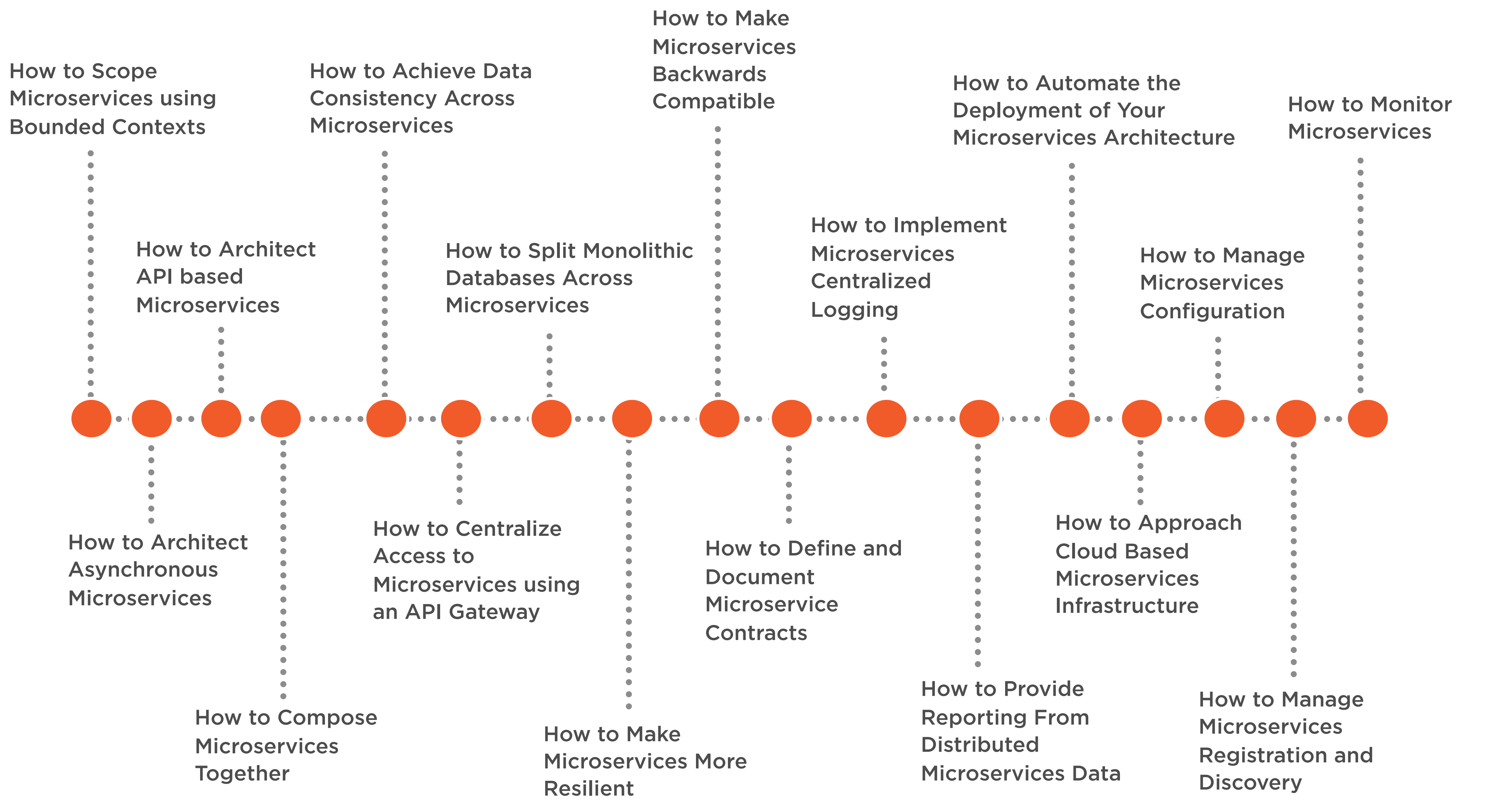
How to Automate the Deployment of Your Microservices Architecture

Rag Dhiman

@ragdhiman www.ragcode.com

Microservices Architectural Design Patterns Playbook


How to Scope Microservices using Bounded Contexts
How to Architect Asynchronous Microservices
How to Architect API based Microservices
How to Compose Microservices Together
How to Achieve Data Consistency Across Microservices
How to Centralize Access to Microservices using an API Gateway
How to Split Monolithic Databases Across Microservices
How to Make Microservices More Resilient
How to Make Microservices Backwards Compatible
How to Define and Document Microservice Contracts
How to Implement Microservices Centralized Logging
How to Provide Reporting From Distributed Microservices Data
How to Automate the Deployment of Your Microservices Architecture
How to Approach Cloud Based Microservices Infrastructure
How to Manage Microservices Configuration
How to Manage Microservices Registration and Discovery
How to Monitor Microservices

Microservices Architectural Design Patterns Playbook

Microservices Architecture

Rag Dhiman

@ragdhiman www.ragcode.com

Microservices Architectural Design Patterns Playbook

Rag Dhiman

@ragdhiman www.ragcode.com

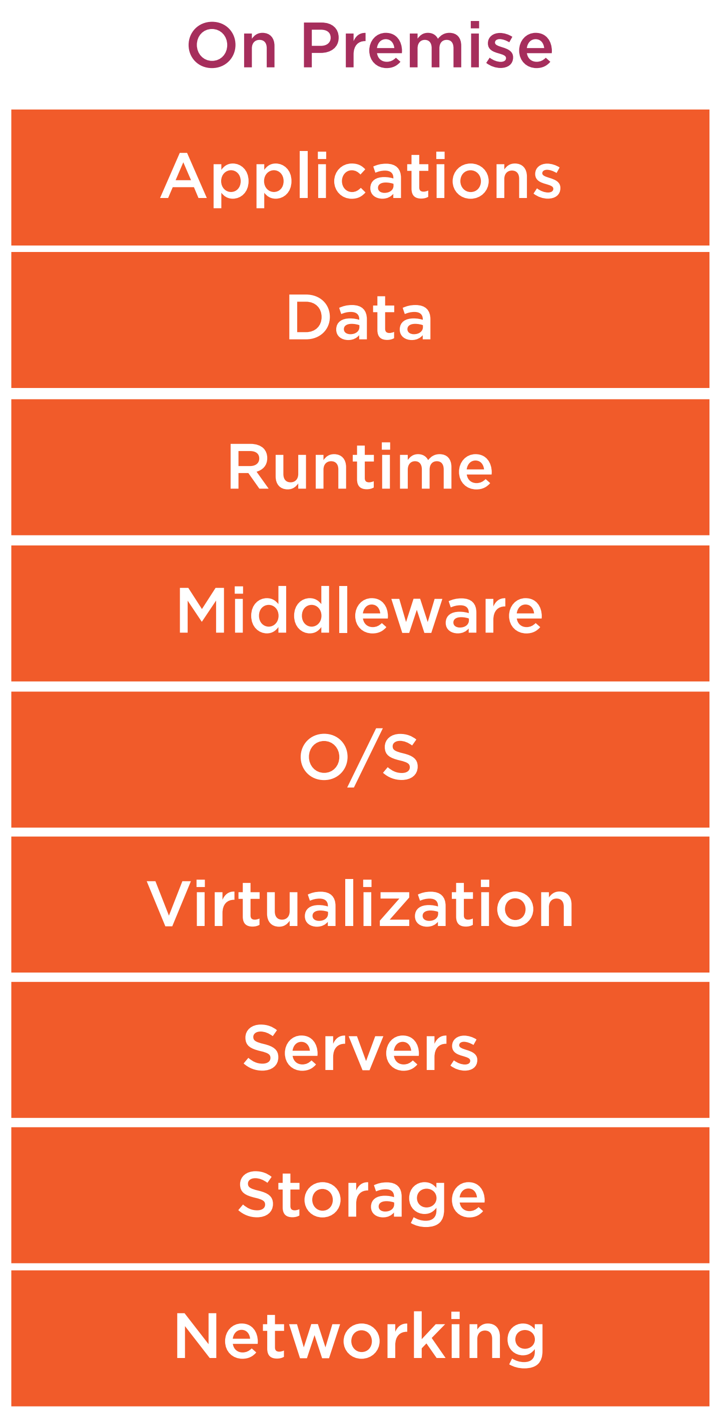
Overview

Continuous Integration Tool

Continuous Delivery Tool

Automation High-level

Introduction



Microservices results in more components

- Complicates building
- Complicates testing
- Complicates packaging
- Complicates deployment

A manual approach is prone to errors

Use automation tools to simplify

Use continuous integration tools

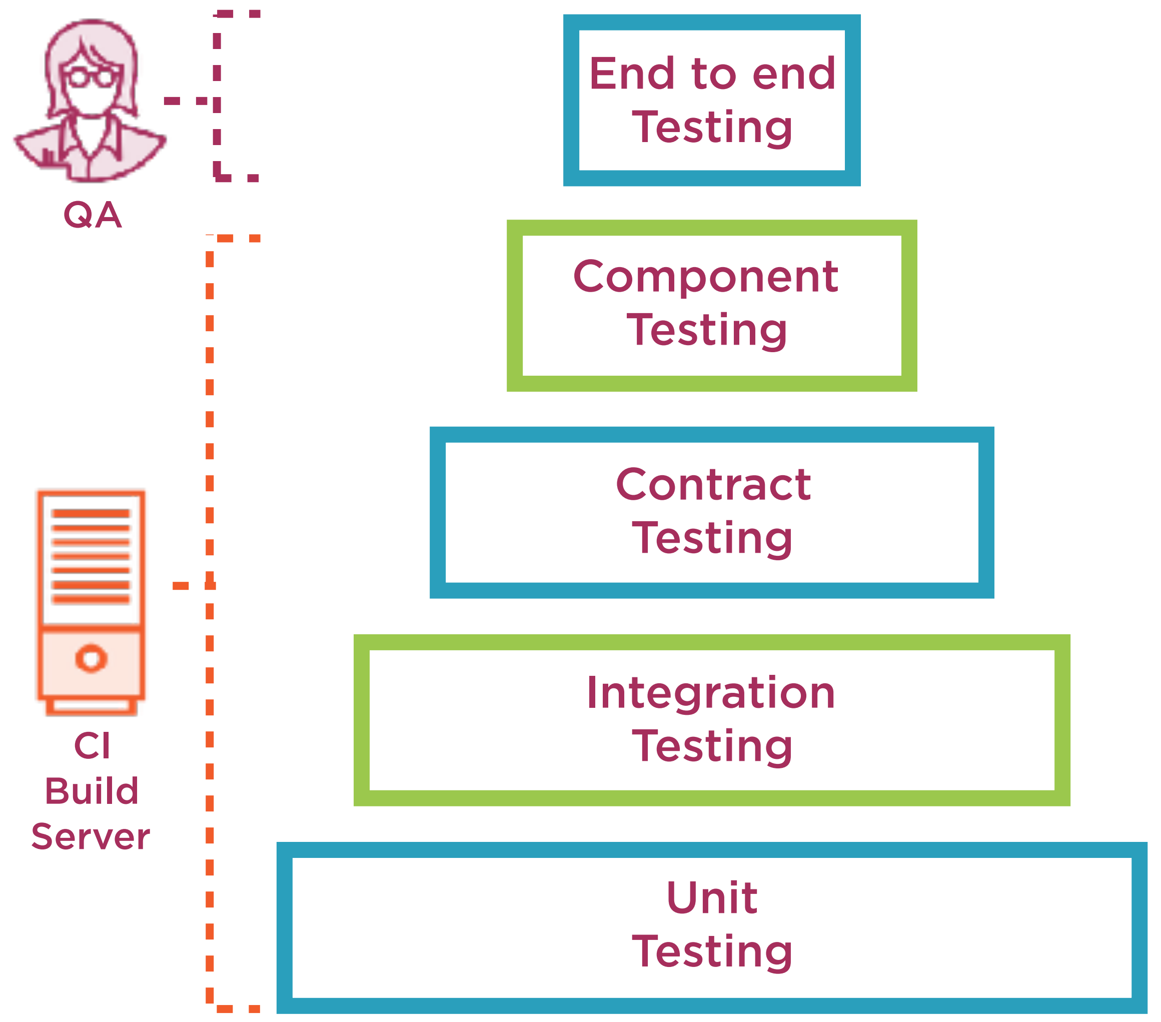
- Automate builds, testing and packaging

Use continuous delivery tools

- Automate software deployment

Continuous Integration

Continuous Integration Tool



Integrating code frequently

- Central code repository

Prevent problems

- From merges, breaking changes and conflicts

Build servers for builds, tests and deployment

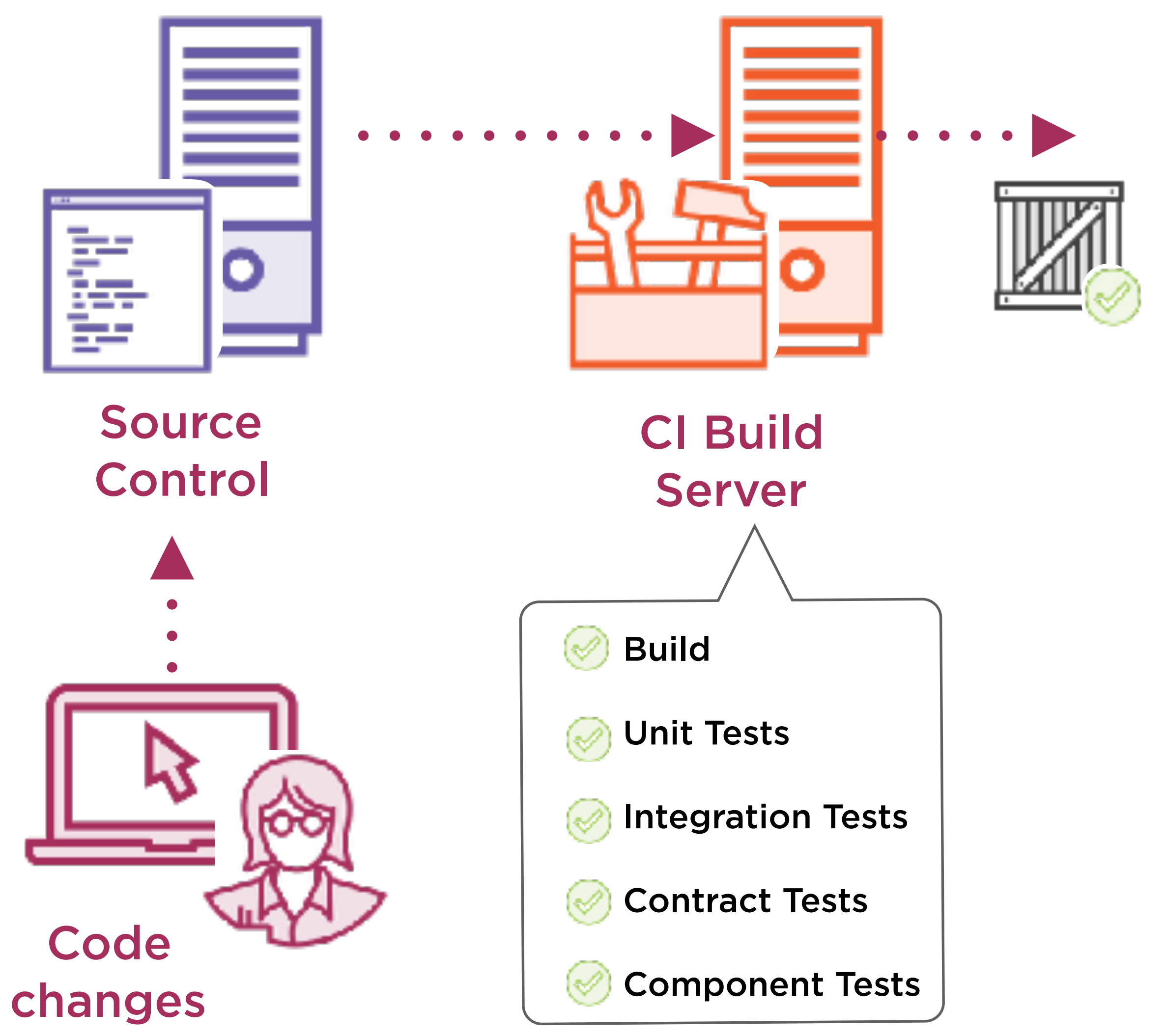
Importance for microservices

- More components = more to test
- Reduce work for QA
- Immediate feedback on breaking changes
- Progress to production with confidence

Many tools with cloud compatibility

- TeamCity, Jenkins, Codeship

Continuous Integration Tool Process



Central source control system is used

- Central code repository

Local copy of code taken

- Creates local code repository

Local code changes are committed

- Commit to local repository

Local code changes merged to central repo

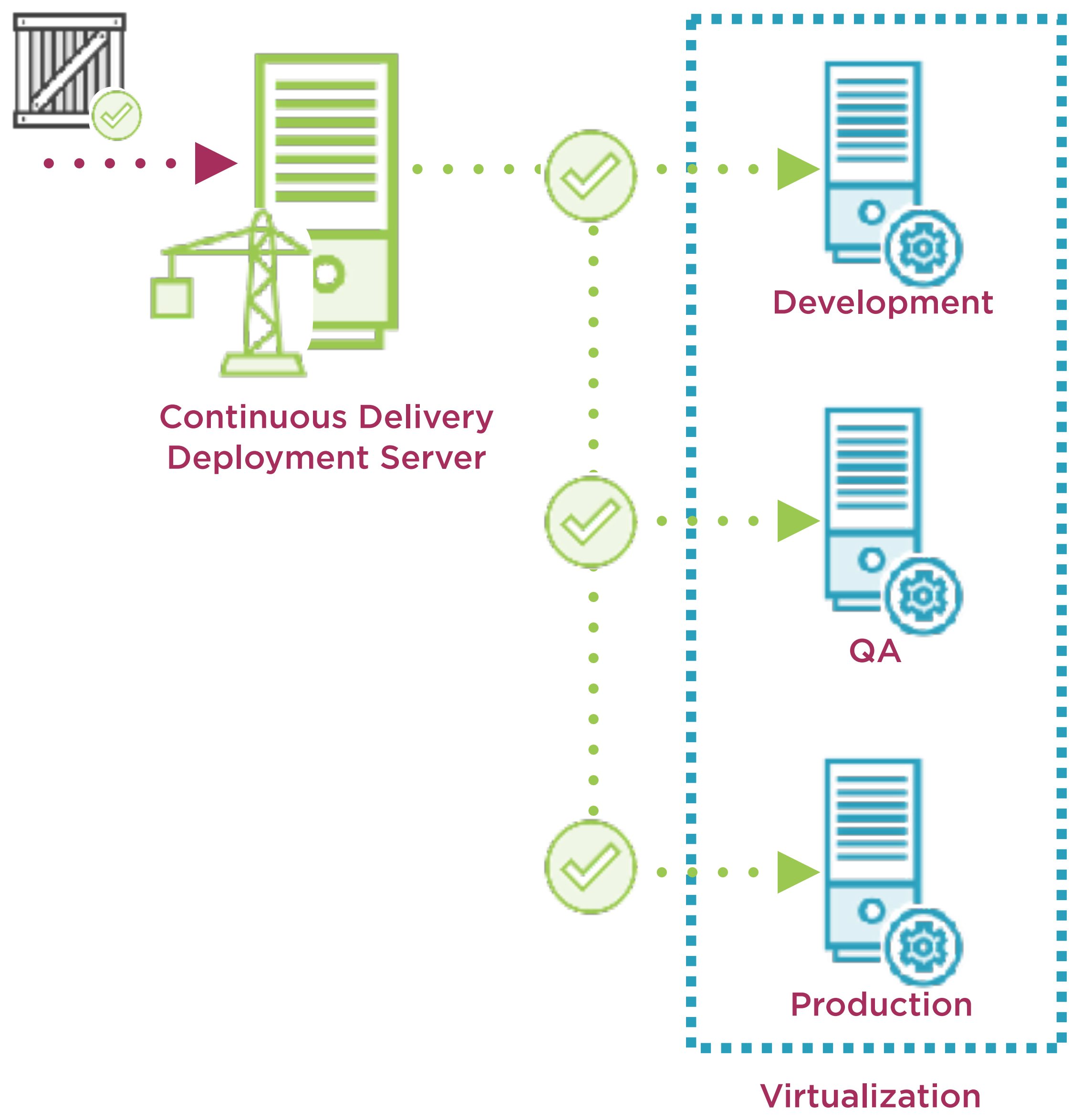
- Pull, resolve merge issues and push

Build server builds, tests and deploys

- Review feedback from build server

Continuous Delivery

Continuous Delivery Tool



Releasable software produced in short cycles

- Develop, test and release faster
- Reduce cost, time and risk

Use continuous delivery tools

- Deployment server
- Environment and release configuration

Deployment pipeline with validation gates

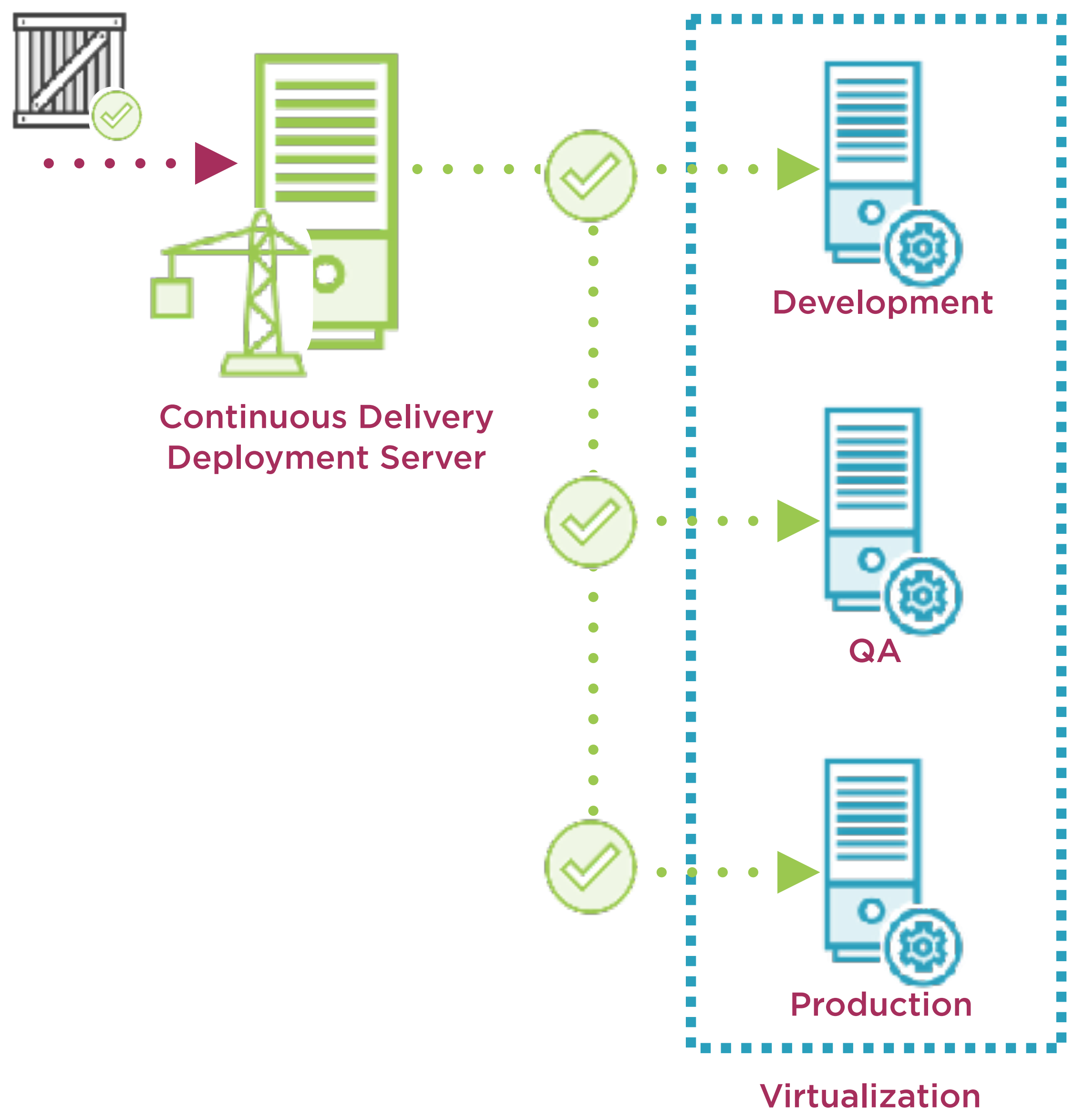
Importance for microservices architecture

- Microservices are ideal for frequent releases
- Reliable release for complex architecture

Many tools with cloud compatibility

- Octopus Deploy, Jenkins

Continuous Delivery Tool Process



Code is compiled and packaged

- Using a CI build server

The build is received by the CD tool

- Deployment server

Deployment server has configuration

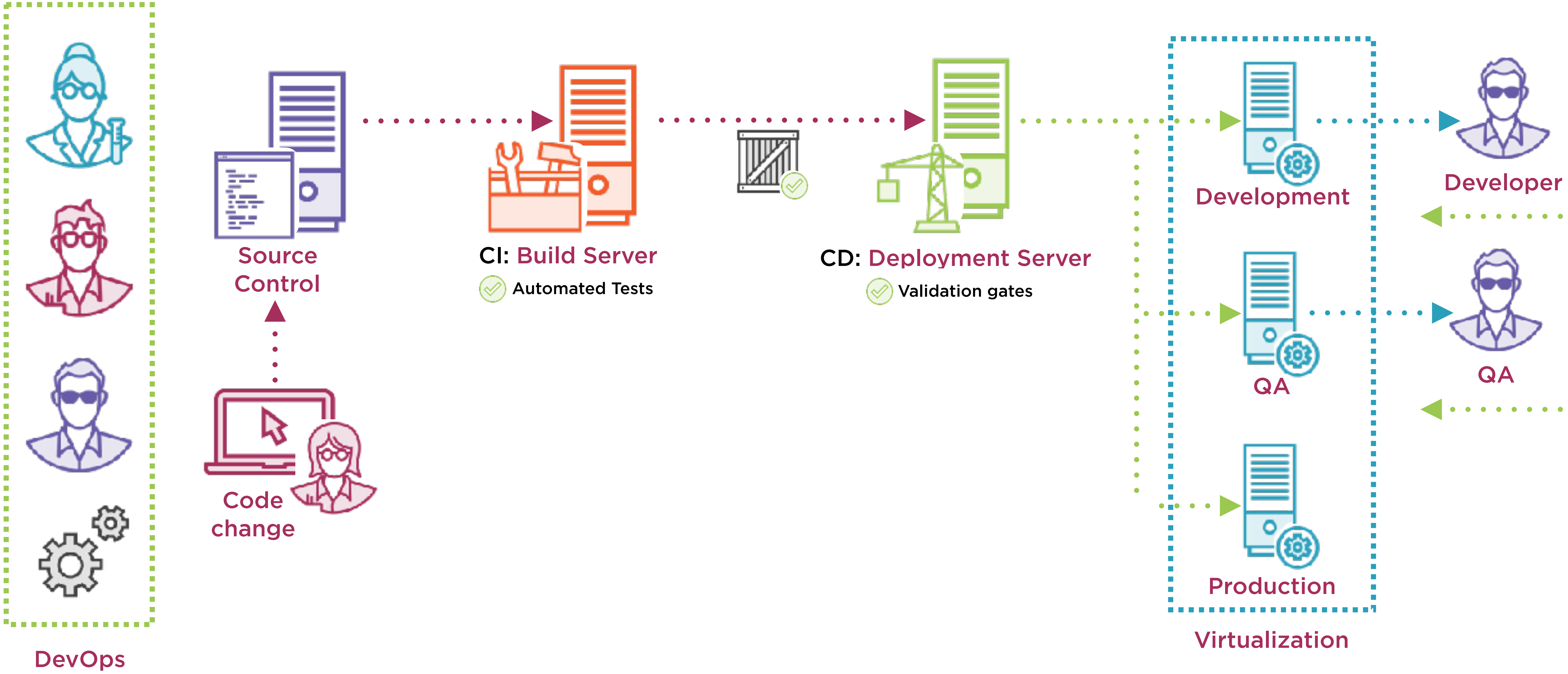
- Environment configuration
- Release configuration

Deployment server uses validation gates

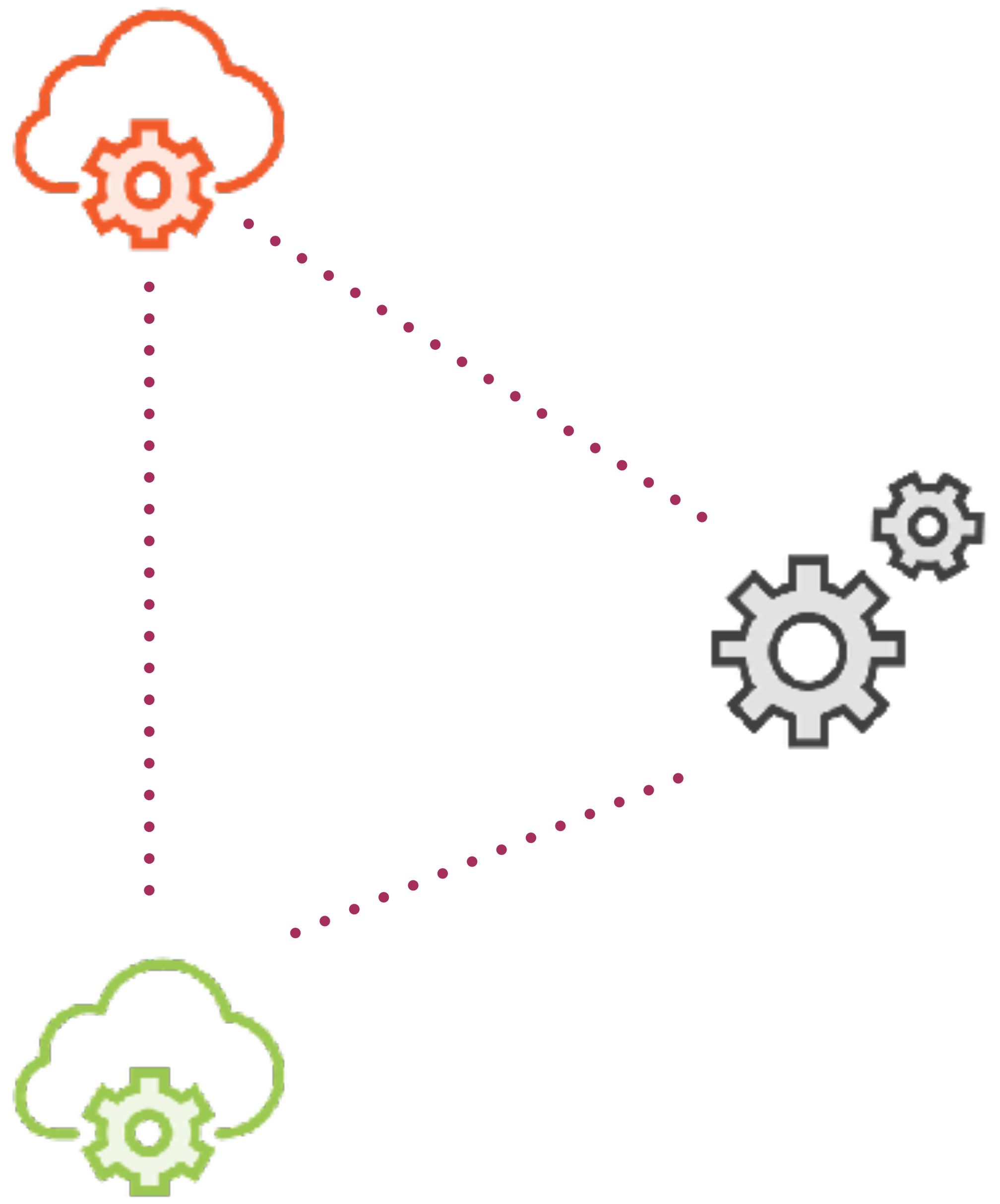
- Development before QA
- QA before production

Automation High-level

Automation High-level



Automation Tools Summary



Source control
- VSTS, GitHub, Jira

Build servers
- TeamCity, Jenkins, Codeship

Deployment servers
- Octopus Deploy, Jenkins

Ad-hoc tasks
- PowerShell, Python

Cloud providers
- Azure, AWS, Google Cloud

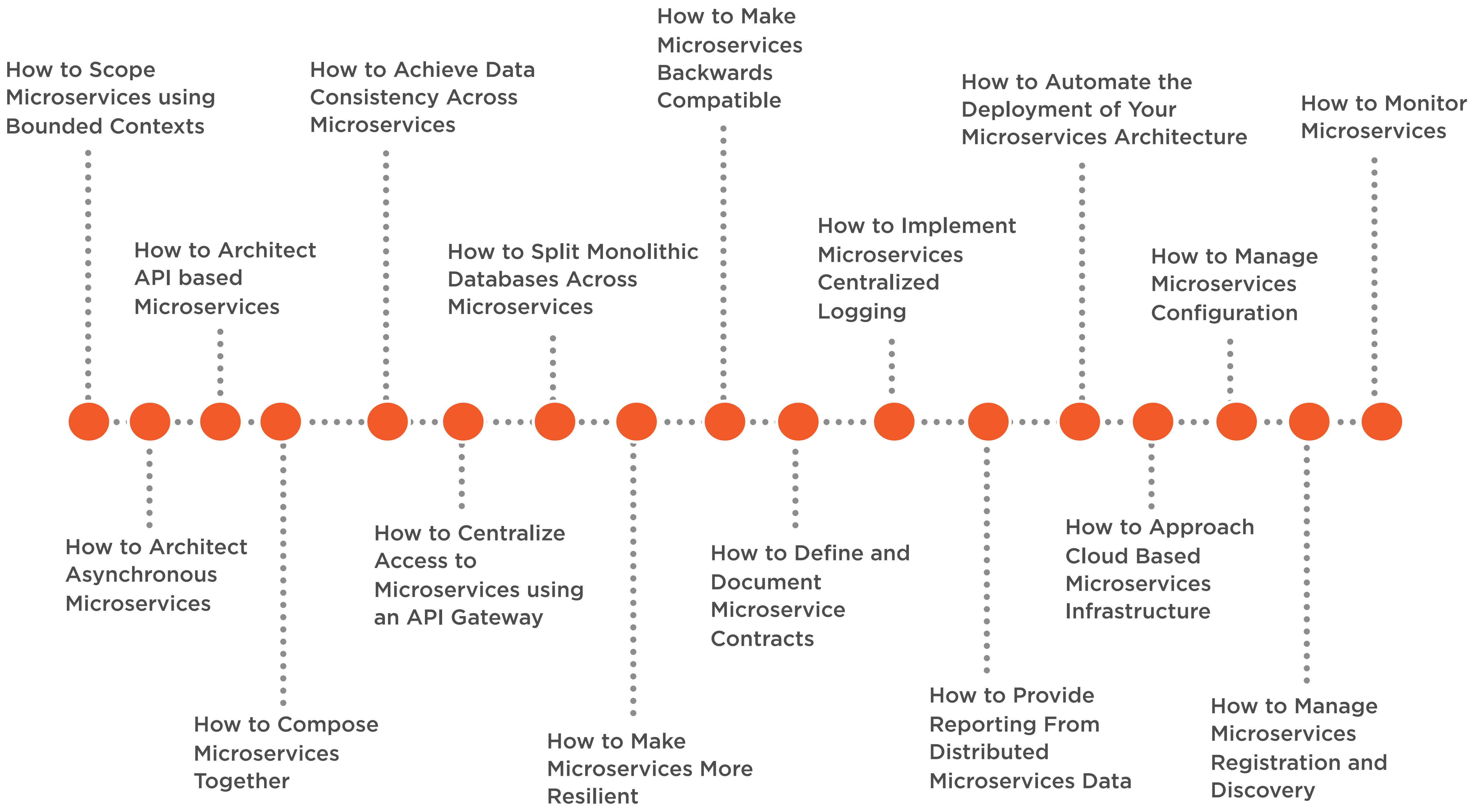
Summary

Continuous Integration Tool

Continuous Delivery Tool

Automation High-level

Microservices Architectural Design Patterns Playbook


How to Scope Microservices using Bounded Contexts
How to Architect Asynchronous Microservices
How to Architect API based Microservices
How to Compose Microservices Together
How to Achieve Data Consistency Across Microservices
How to Centralize Access to Microservices using an API Gateway
How to Split Monolithic Databases Across Microservices
How to Make Microservices More Resilient
How to Make Microservices Backwards Compatible
How to Define and Document Microservice Contracts
How to Implement Microservices Centralized Logging
How to Provide Reporting From Distributed Microservices Data
How to Automate the Deployment of Your Microservices Architecture
How to Approach Cloud Based Microservices Infrastructure
How to Manage Microservices Configuration
How to Manage Microservices Registration and Discovery
How to Monitor Microservices