IBM Training

Zero Downtime Deployment

Deploying a new version of an application to replace an old version without incurring a service outage

- Characteristics of zero downtime deployment
  - Application is always available, first old version, then new version
  - User does not experience an outage (a.k.a. downtime)
  - Old version and new version are both deployed at the same time – traffic is directed to both
  - User's only indication of new deployment is improved functionality
- Necessitated by DevOps' practice of frequent redeployment
  - Unacceptable for each new deployment to cause an outage

IBM Training

Implementing Zero Downtime Deployment – Blue Green Deployment

- Some challenges of continuous deployment
  - Application availability during cut-over
  - Rapidly reverting to prior version if necessary
- A zero-downtime deployment technique
  - Also known as Red Black Deployment (Netflix term)
- Two nearly identical production environments, called Blue and Green
  1. Deploy v1 (blue environment), which users use
  2. Deploy v2 (green environment)
  3. Run smoke tests and automated tests to verify the new environment before taking it live
  4. Shift client load from Blue to Green
  5. Delete Blue (or keep as a hot backup)
- Kubernetes incorporates "rolling updates" as its default method of deployment
  - Updates are incrementally rolled out pod by pod, so some pods have the "old" version and some have the "new" version – traffic can be managed between these pods
  - Updates can be rolled back if there is a problem

"Blue green deployments"
https://www.ng.bluemix.net/docs/manageapps/updapps.html

IBM Training

DevOps in Twelve-Factor Apps

IBM Training

The Twelve-Factor App

- A set of best practices for creating applications
  - Implementing, deploying, monitoring, and managing
- Typical modern applications
  - Deployed in the cloud
  - Accessible as web applications that deliver software-as-a-service
- Can be applied to any application
  - Implemented in any programming language
  - Using any backing services (database, messaging, caching, and so on)
- Addresses common problems
  - The dynamics of the organic growth of an app over time
  - The dynamics of collaboration between developers
  - Avoiding the cost of software erosion
  - Systemic problems in modern application development
- Provides a shared vocabulary for addressing these problems

Sources: http://www.12factor.net
http://www.clearlytech.com/2014/01/04/12-factor-apps-plain-english/

IBM Training

Twelve Factors related to DevOps

I. **Codebase**: One codebase that is tracked in revision control, with many deployments
II. **Dependencies**: Explicitly declare and isolate dependencies
III. **Configuration**: Store Configuration in the environment
IV. **Backing services**: Treat backing services as attached resources
V. **Build, release, run**: Strictly separate build and run stages
VI. **Processes**: Execute the app as one or more stateless processes
VII. **Port binding**: Export services with port binding
VIII. **Concurrency**: Scale out using the process model
IX. **Disposability**: Maximize robustness with fast startup and efficient shutdown
X. **Development and production parity**: Keep development, staging, and production as similar as possible
XI. **Logs**: Treat logs as event streams
XII. **Admin processes**: Run administrative and management tasks as one-off processes

IBM Training

Twelve Factors related to DevOps

I. **Codebase**: One codebase that is tracked in revision control, with many deployments
II. **Dependencies**: Explicitly declare and isolate dependencies
III. **Configuration**: Store Configuration in the environment
IV. **Backing services**: Treat backing services as attached resources
V. **Build, release, run**: Strictly separate build and run stages
VI. **Processes**: Execute the app as one or more
VII. **Port binding**: Export services with port binding
VIII. **Concurrency**: Scale out using the process model
IX. **Disposability**: Maximize robustness with fast startup and efficient s
X. **Development and production parity**: Keep development, staging, and production as similar as possible
XI. **Logs**: Treat logs as event streams
XII. **Admin processes**: Run administrative and management tasks as one-off processes

- Delivery pipeline is triggered by SCM such as Git repos
- Delivery pipeline uses environment variables, both system and user
- Delivery pipeline builds deployment artifacts, packages immutable images, and deploys to lifecycle stages (e.g. test, stage, prod)
- Delivery pipeline deploys same immutable image to each lifecycle stage

IBM Training

Implementing DevOps

IBM Training

Transformation to DevOps Culture and Tooling

- Culture change is key to successful transformation from top to bottom
- Small integrated teams
  - rapidly iterating and experimenting
  - deploying new versions many times per day
  - guided by measurements
- Key competitive advantage

=> And how do you **roll this out at scale** to teams across your organization?

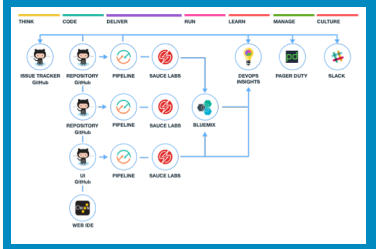
IBM Training

Implementing DevOps is not all about technology

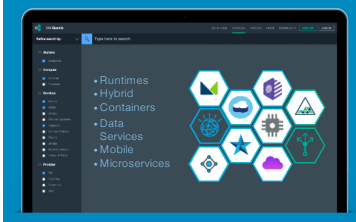
You need new practices and you need tools

Method
IBM Cloud Garage Method

Integrated Toolchain
IBM Cloud Continuous Delivery



Innovation Platform
IBM Cloud



IBM Training

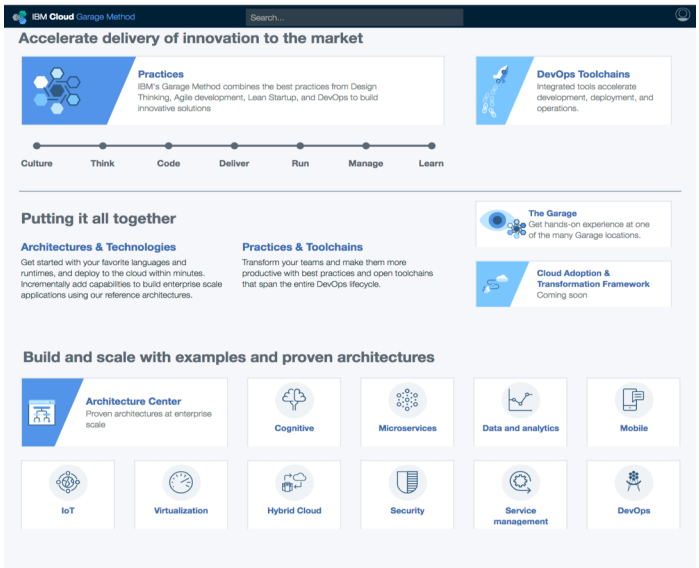
Method - IBM Cloud Garage Method

ibm.com/cloud/garage

Combining Industry Best Practices for Design Thinking, Lean Startup, Agile Development, DevOps and Cloud to build and deliver innovative solutions

Reference architectures show how services work together, with sample implementations

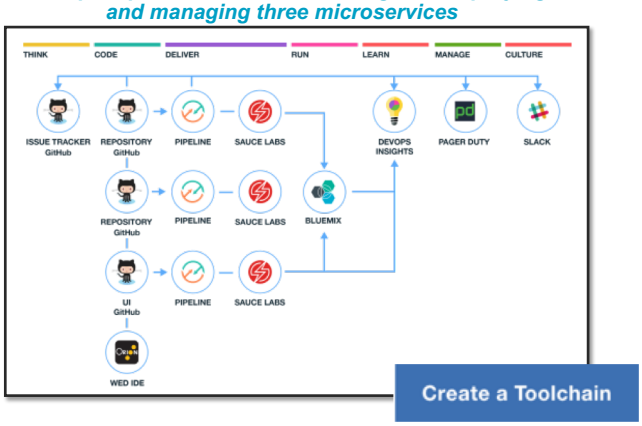
Toolchains support the practices and architectures



IBM Training

Integrated Toolchain - Built on the Open Toolchain foundation

A sample open toolchain for building, and deploying and managing three microservices



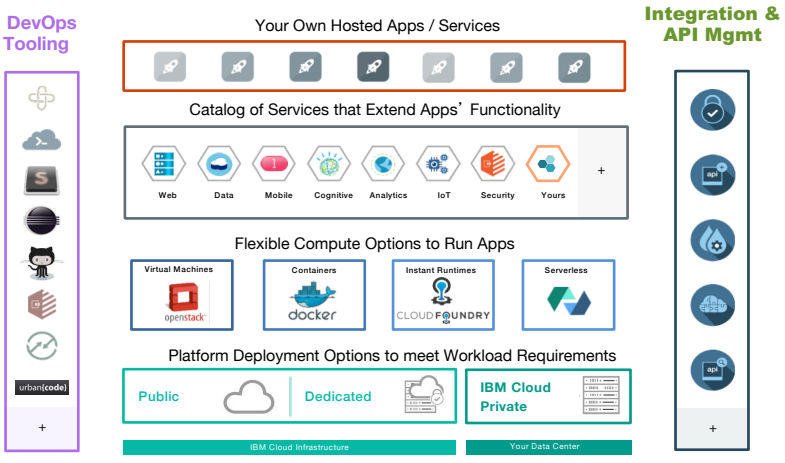
Toolchains provide an integrated set of tools that support the best practices to build, deploy and manage your apps

You can create toolchains that include IBM Cloud services, open source tools, and third-party tools that make development and operations repeatable and easier to manage

Rapidly instantiate new toolchains from templates to on-board new teams quickly

IBM Training

Innovation Platform - IBM Cloud



IBM Training

Conclusion

IBM Training

Summary

- What is DevOps
- Definitions - CI / CD
- Zero Downtime Deployment
- DevOps in Twelve-Factor Apps
- Implementing DevOps

IBM Training

Jenkins in IBM Cloud Private

IBM Training

Objectives

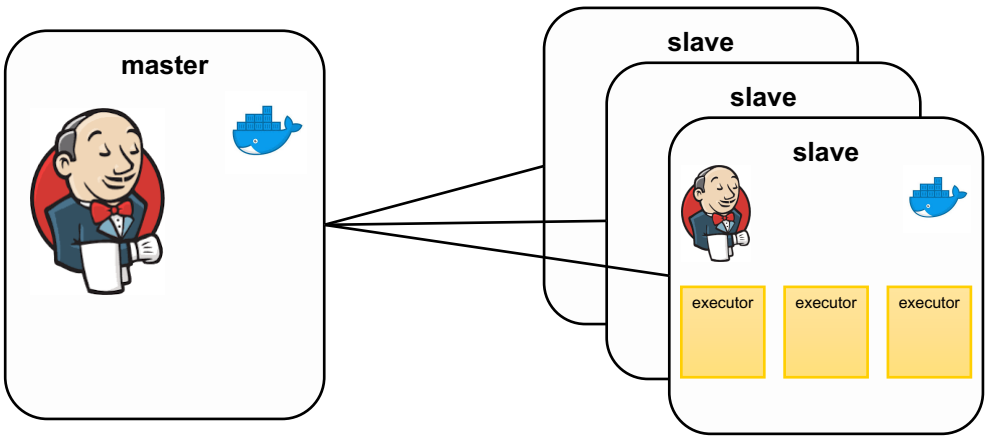
- Describe Jenkins components and architecture
- Describe how the Jenkins interact with Git
- Explain what a Jenkins pipeline is
- Build pipeline for build and deployment in IBM Cloud Private

IBM Training IBM

Jenkins components

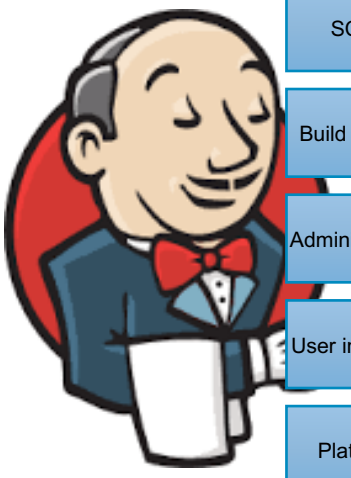
IBM Training IBM

Jenkins components



IBM Training

Jenkins plugins

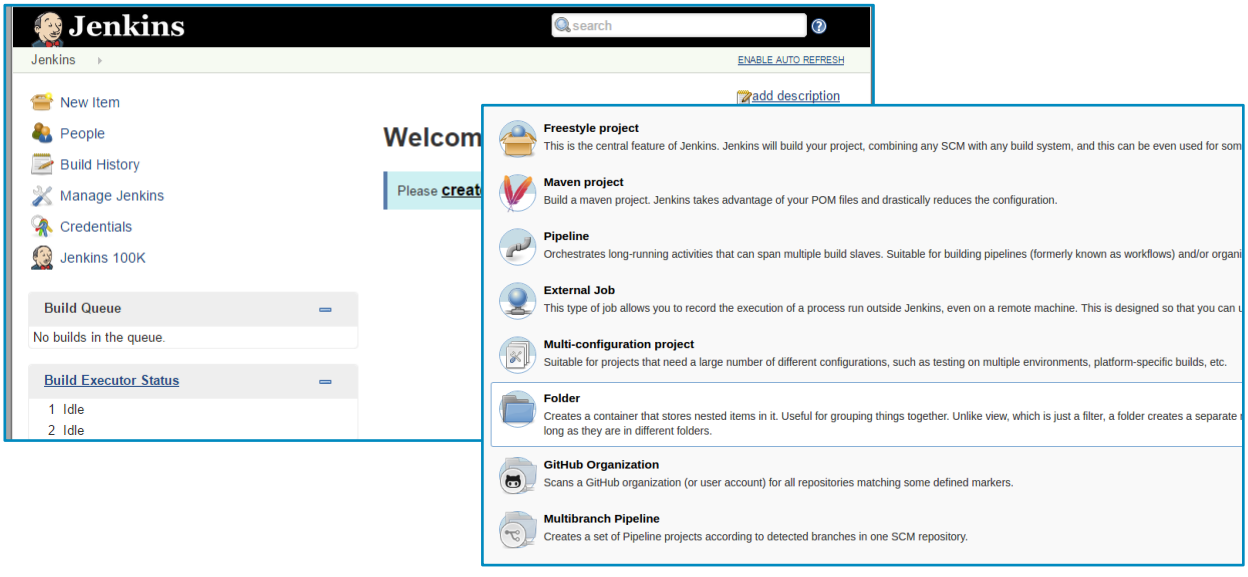


SCM plugin	Interact with SCM repository to check for updates and invokes pipelines
Build action plugin	Deployment tools Build actions and notification
Administration plugin	Controlling agents, user security CLI extensions, cluster management
User interface plugin	Modify the Jenkins Web UI widgets Custom column displays
Platform plugin	Development target platform that provide custom build processing: iOS, Android, Ruby, .NET, etc

Jenkins in ICP 5 © Copyright IBM Corporation 2018

IBM Training

Jenkins Web UI



Jenkins in ICP 6 © Copyright IBM Corporation 2018

IBM Training

Pipelines and stages

IBM Training

Jenkins and SCM

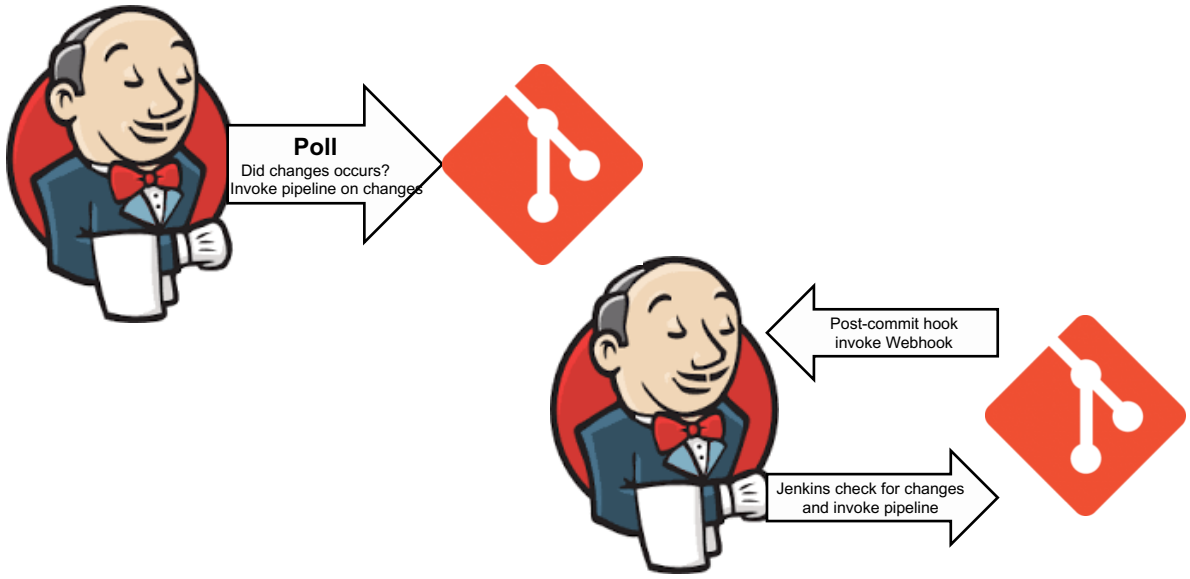
IBM Training

Jenkins SCM plugin

- Allows Jenkins to interact with SCM
  - Check for any changes in SCM
  - Pull SCM content
  - Get change log from SCM
- Available for GIT, github, BitBucket, SVN and others
- Provides a trigger Webhook for SVN to trigger checking

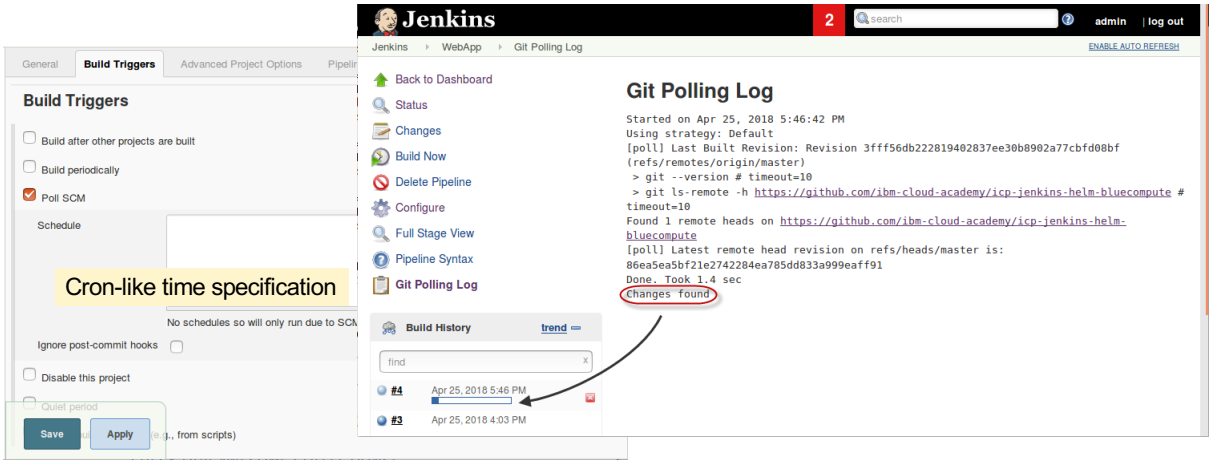
IBM Training

Jenkins SCM interaction modes



IBM Training

Jenkins poll to SCM



IBM Training

Jenkins and GIT

1. The post-commit hook in .git/hooks is invoked when git push command is issued.
2. The post-commit triggers Jenkins webhook to check the SCM (web hook is created when Poll SCM box is checked)
3. As this is triggered by post-commit hook, changes that is found invokes the corresponding build pipeline

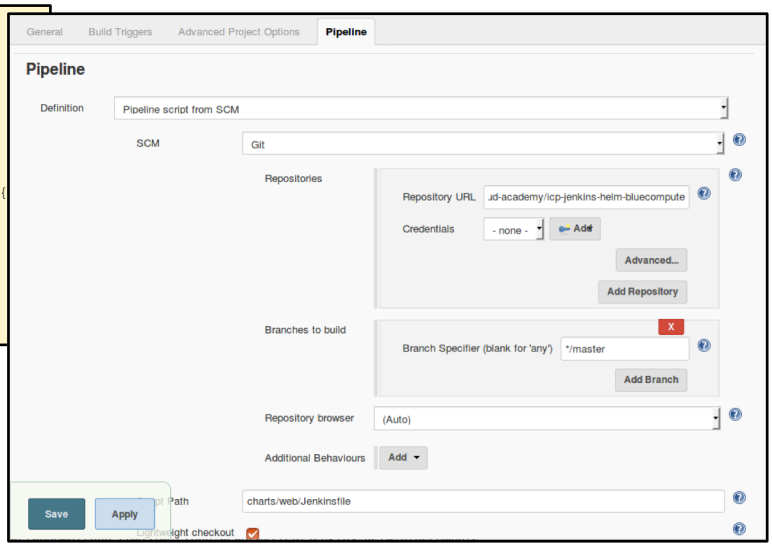
IBM Training

Pipeline definition

```
{
    node ('mypod') {
        checkout scm
        container('cli') {
            stage('Build Gradle Project')
                sh """

                """
            }
            stage ('Build Docker Image') {
                sh """

                """
            }
        }
    }
}
```

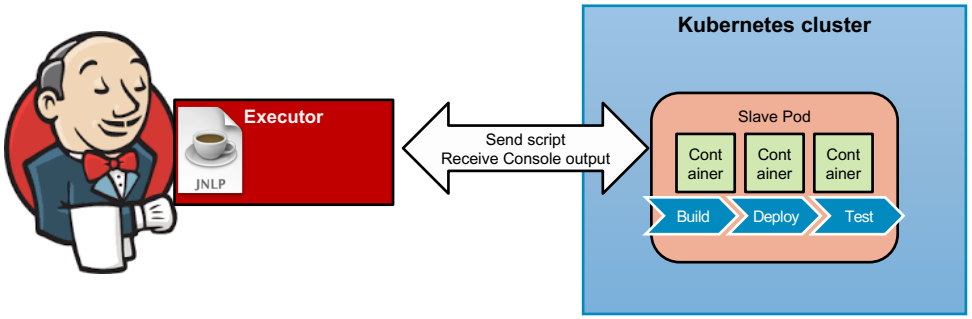


IBM Training

Kubernetes plugin and pipelines

IBM Training

Jenkins Kubernetes plugin



Kubernetes plugin allows running of Jenkins pipeline in a Kubernetes cluster
The pipeline would run on a slave pod which is built based on a podTemplate
The slave pod is controlled by a Jenkins executor (which is a JNLP process)

Jenkins in ICP 15

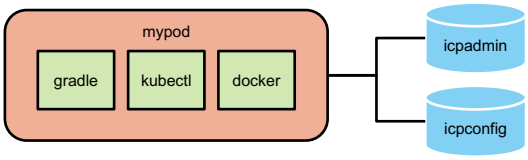
IBM Training

Pipeline podTemplate

```
podTemplate(
  label: 'mypod',
  volumes: [
    hostPathVolume(hostPath: '/var/run/docker.sock', mountPath: '/var/run/docker.sock'),
    secretVolume(secretName: 'icpadmin', mountPath: '/var/run/secrets/registry-account'),
    configMapVolume(configMapName: 'icpconfig', mountPath: '/var/run/configs/registry-config')
  ],
  containers: [
    containerTemplate(name: 'gradle', image: 'ibmcase/gradle:jdk8-alpine', ttyEnabled: true, command: 'cat'),
    containerTemplate(name: 'kubectl', image: 'ibmcloudacademy/k8s-icp:v1.0', ttyEnabled: true, command: 'cat'),
    containerTemplate(name: 'docker' , image: 'docker:17.06.1-ce', ttyEnabled: true, command: 'cat')
  ]
)
```

```
apiVersion: v1
kind: Secret
metadata:
  name: icpadmin
type: Opaque
data:
  username: <user-encoded>
  password: <password-encoded>
```

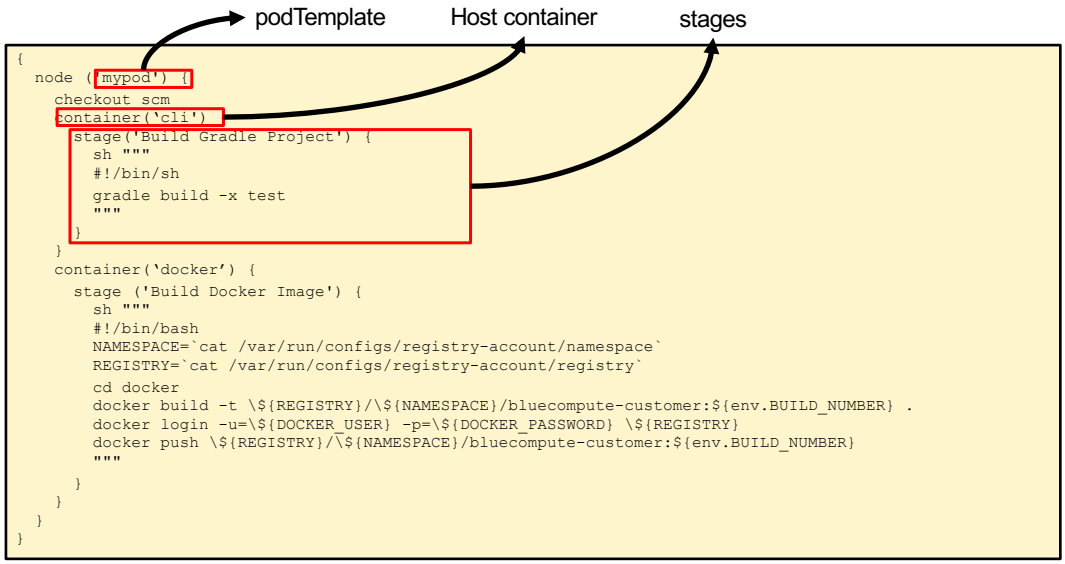
```
apiVersion: v1
kind: ConfigMap
metadata:
  name: icpconfig
data:
  namespace: default
  registry: cloudcluster.icp:8500
```



Jenkins in ICP 16

IBM Training

Jenkins pipeline in Kubernetes

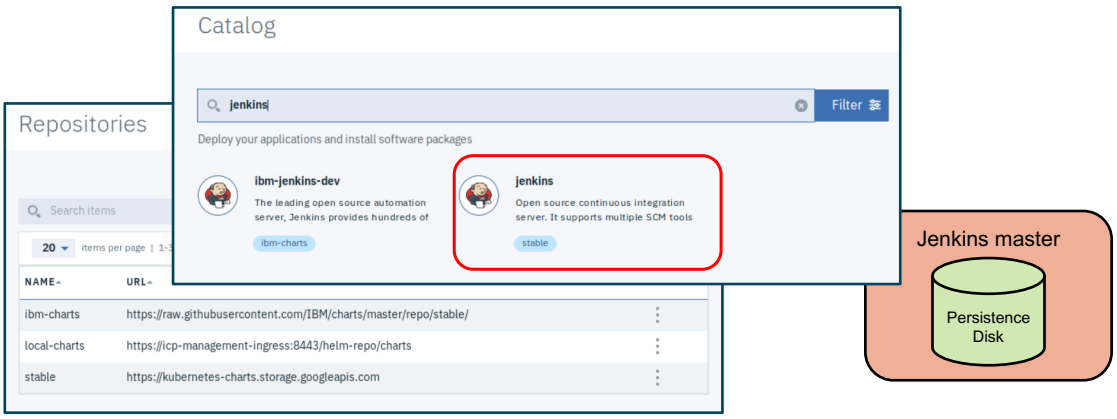


IBM Training

Jenkins in IBM Cloud Private

IBM Training

Jenkins in IBM Cloud Private



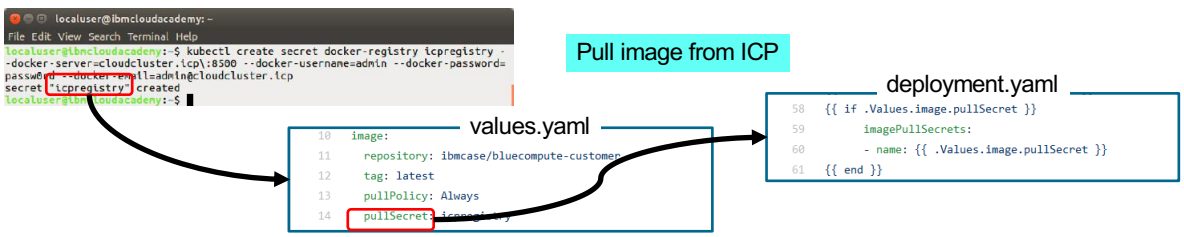
https://github.com/kubernetes/charts/tree/master/stable/jenkins

IBM Training

Jenkins pipeline access to IBM Cloud Private repository

```
stage ('Push Docker Image to Registry') {
  sh """
  #!/bin/bash
  NAMESPACE=`cat /var/run/configs/registry-config/namespace`
  REGISTRY=`cat /var/run/configs/registry-config/registry`
  DOCKER_USER=`cat /var/run/secrets/registry-account/username`
  DOCKER_PASSWORD=`cat /var/run/secrets/registry-account/password`
  docker login -u=\${DOCKER_USER} -p=\${DOCKER_PASSWORD} \${REGISTRY}
  docker push \${REGISTRY}/\${NAMESPACE}/bluecompute-customer:${env.BUILD_NUMBER}
  """
}
```

Push image to ICP



IBM Training

Jenkins deploying to IBM Cloud Private

```
container('kubectl') {
  stage('Deploy new Docker Image') {
    sh """
    #!/bin/bash
    set +e
    NAMESPACE=`cat /var/run/configs/registry-config/namespace`
    REGISTRY=`cat /var/run/configs/registry-config/registry`
    DOCKER_USER=`cat /var/run/secrets/registry-account/username`
    DOCKER_PASSWORD=`cat /var/run/secrets/registry-account/password`
    wget --no-check-certificate https://master:8443/api/cli/icp-linux-amd64
    bx plugin install icp-linux-amd64
    bx pr login -a https://master:8443 --skip-ssl-validation \
      -u \${DOCKER_USER} -p \${DOCKER_PASSWORD} -c id-cloudcluster-account
    helm init --client-only
    bx pr cluster-config cloudcluster
    helm repo add bluecompute \
      https://raw.githubusercontent.com/ibm-cloud-academy/icp-jenkins-helm-bluecompute/master/charts
    helm install --tls -n bluecompute-customer \
      --set image.repository=\${REGISTRY}/\${NAMESPACE}/bluecompute-customer \
      --set image.tag=${env.BUILD_NUMBER} bluecompute/customer
    """
  }
}
```

Jenkins in ICP 21

IBM Training

Monitoring IBM Cloud Private

IBM Training

Agenda

- Monitoring approaches in IBM Cloud Private
- What is available out of the box
- Cloud Service Management and Operations

IBM Training

Who monitors?

Primary Role

Todd

Operations / Admin

Responsible for infrastructure, security, and management of the environment

Todd has confidence in the private cloud platform. He knows corporate data is secure and in compliance with government regulations and/or industry standards. He has control and visibility of the running applications and can quickly update workloads and the underlying platform using continuous delivery techniques, without requiring downtime. Application data is automatically managed for backup, recovery, and disaster failover into alternate locations.

Todd receives incident alerts
He ensures monitoring is in place for the application components

Secondary Role

Jane

Enterprise Developer

Responsible for modernizing existing applications and creating new Cloud Native Workloads

Jane can immediately leverage CI/CD to rapidly develop, test and deliver her applications on the IBM cloud platform. She has a core set of services she needs, including data and application runtime services, with prescriptive guidance for moving heritage workloads to the cloud and/or creating new cloud native applications. She can also safely integrate off-premises cloud services into her application.

Jane leverages build to manage so that Todd can deploy "health" tests against the APIs.
She is concerned with her applications performance and trends

Monitoring IBM Cloud Private 3

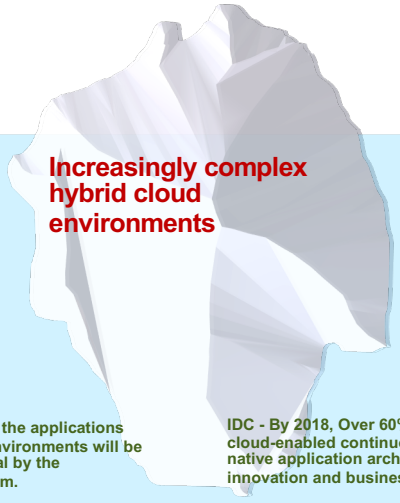
IBM Training

Challenges with today's IT landscapes – yet another reality!

…Managing today's applications is difficult because symptoms may not be visible

Mean Time to Repair for application-related problems is often **3-6 hours**

User calls to applications support are the enterprise's "heads up" for an application problem **35% of the time**



Issues that go beyond Level 1 support often involve at least **3-4 people** taking up an average of **5-7 total man hours**

58% of survey participants said their enterprise had between **6 and 40 different monitoring tools**

Gartner – By 2018, 50% of the applications running in public cloud environments will be considered mission-critical by the organizations that use them.

IDC - By 2018, Over 60% of new apps will use cloud-enabled continuous delivery and cloud-native application architectures to enable faster innovation and business agility.

IDC FutureScape: Worldwide Cloud 2016 Predictions -Mastering the Raw Material of Digital Transformation, IDC, Nov. 3, 2015

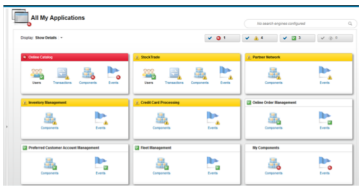
Gartner Predicts 2016: Cloud Computing to Drive Digital Business 12/8/15 G00292047

Monitoring IBM Cloud Private 4

IBM Training

Primary purpose for monitoring and logging

- Overall system health & availability
  - Understand, maintain, prevent, diagnose issues related to availability. What's broken and why?
- Business relevant data
  - Understand application & infrastructure usage patterns
  - Improve customer experience, test and measure hypothesis (A/B testing for example)
  - Optimize resources for greater economic efficiency



IBM Training

Traditional and Cloud-native

- Cloud-native has the same monitoring characteristics as traditional (non-Cloud-native)

- Infrastructure monitoring covering hosts and the network
- Application monitoring covering service level data like errors or response times
- Real-user monitoring covering end-user response times and errors as well behavioral data
- Log monitoring

IBM Training

Monitoring and logging

- Monitoring helps identify and solve immediate problems such as:
  - Code problem
  - Deployment problem
  - Response time issue
  - Application availability
  - Platform issue
  - Typically numeric time series data used to record state that changes frequently
- Logging is used
  - Log files
    - Used to record important events, such as server start/stop, authentication results, a new version of the service is deployed, unexpected errors, application start/stop, configuration changes, etc.
  - Critical for effective debugging
  - To identify longer term trends and problems
    - Resource issues over time
      - Does an application consume more memory at certain times of day or season?
      - Is more/less hardware needed based on resource consumption usage?
    - Are workloads meeting Service Level Agreements?

IBM Training

Operational Visibility

- Alerts
  - Used to automatically detect changes in monitored data relative to a threshold or other condition
  - CPU >90% for 3 minutes, disk volume > 70% full, response latency > 100ms for 50% of requests, etc.
  - Not all alerts may trigger a notification
- Notifications
  - Communicate an important alert has been triggered and needs attention
  - Could be text, Slack post, PagerDuty, sirens, flashing-lights, etc.
- Dashboards
  - Typically a web app used to visualize a summary of the most important data relative to its users
  - Usually have ability to filter and/or ability to do ad-hoc queries
  - Very common to have multiple dashboards for different audiences (executives, offering managers, SRE/ops, developers, etc.)
    - Very unlikely to have a once-size-fits all dashboard

IBM Training

Differences between traditional and Cloud-Native

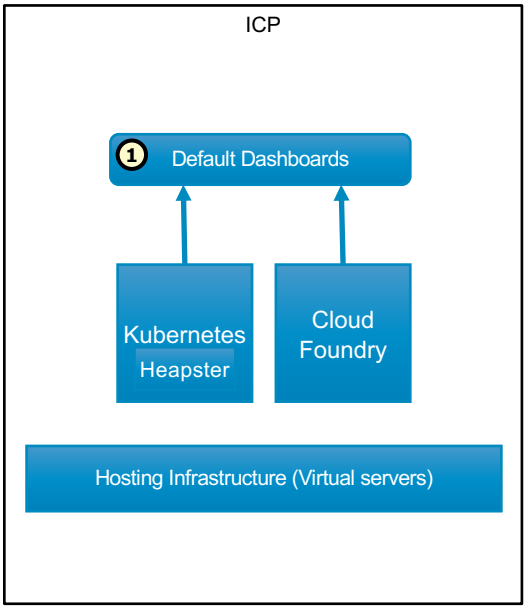
Traditional	Cloud-Native
Centralized operations staff with monitoring and system health responsibility	Development teams have end-to-end responsibility (or share with operations)
Fewer and more controlled releases	Many releases, sometimes daily/hourly
Few language / technology choices	Polyglot language / technology choices (for example, databases)
Anomalies detected from small number of monitored systems	High number of inter-related components making anomaly detection difficult
Monitor and manage at infrastructure level	Monitor and manage at application (workload) level
Pro-active long term capacity planning	Real-time capacity planning with auto-scaling
Fix failing components	Replace failing components

Monitoring IBM Cloud Private 9

IBM Training

IBM Cloud Private Monitoring and Logging

1. The ICP dashboard shows the status, metrics and log entries for all the workloads & Infrastructure.

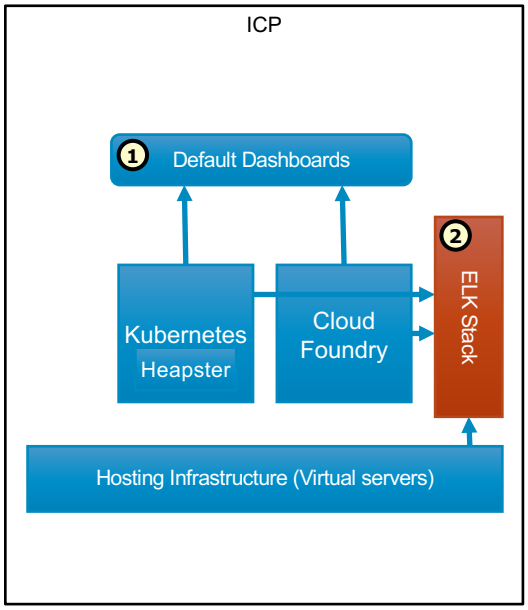


Monitoring IBM Cloud Private 10

IBM Training

IBM Cloud Private Monitoring and Logging

1. The ICP dashboard shows the status, metrics and log entries for all the workloads & Infrastructure.
2. An Elasticsearch stack collects logs from all the workloads and Infrastructure

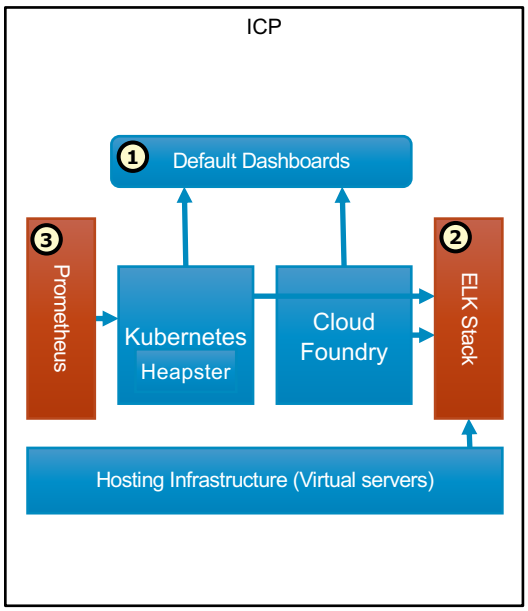


Monitoring IBM Cloud Private 11

IBM Training

IBM Cloud Private Monitoring and Logging

1. The ICP dashboard shows the status, metrics and log entries for all the workloads & Infrastructure.
2. An Elasticsearch stack collects logs from all the workloads and Infrastructure
3. A Prometheus stack collects logs from the Kubernetes workload

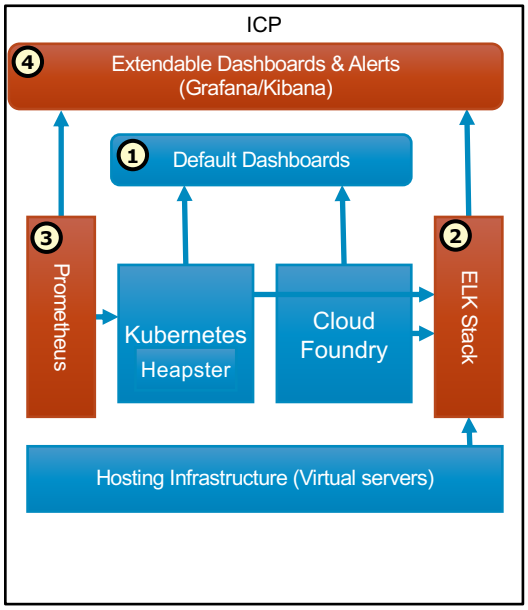


Monitoring IBM Cloud Private 12

IBM Training

IBM Cloud Private Monitoring and Logging

1. The ICP dashboard shows the status, metrics and log entries for all the workloads & Infrastructure.
2. An Elasticsearch stack collects logs from all the workloads and Infrastructure
3. A Prometheus stack collects logs from the Kubernetes workload
4. Customizable Grafana and Kibana dashboards are available for use



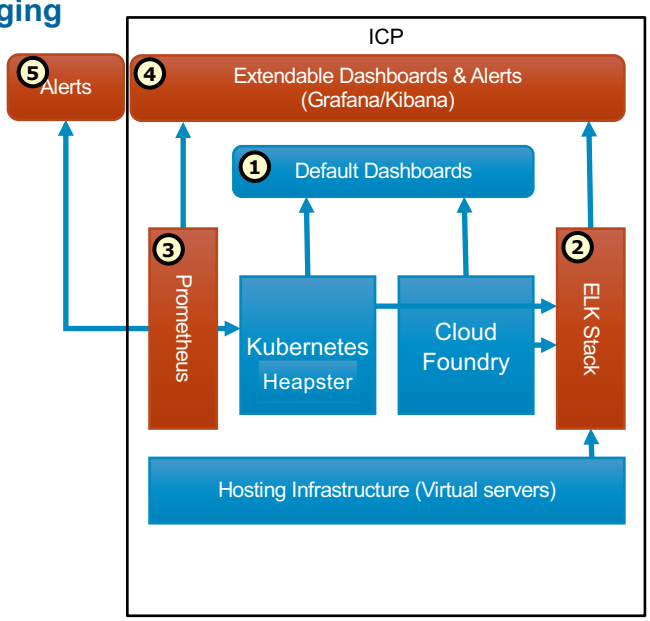
Monitoring IBM Cloud Private 13

IBM Training

IBM Cloud Private Monitoring and Logging

1. The ICP dashboard shows the status, metrics and log entries for all the workloads & Infrastructure
2. An Elasticsearch stack collects logs from all the workloads and Infrastructure
3. A Prometheus stack collects logs from the Kubernetes workload
4. Customizable Grafana and Kibana dashboards are available for use
5. Prometheus can send alerts to external system (email, webhooks)

- Dashboards are view only
  - APIs available to manage



Monitoring IBM Cloud Private 14

IBM Training

Monitoring Components

- Prometheus - https://prometheus.io/
  - Systems and service monitoring system
  - Cloud Native Computing Foundation (CNCF) project
  - Collects metrics from configured targets at given intervals, evaluates rule expressions, displays the results, and can trigger alerts if some condition is observed to be true

- Grafana - https://grafana.com/grafana
  - Open source metric analytics and visualization suite
  - Rich display features (compared to Kibana)
  - Commonly used for visualizing time series data for infrastructure and application analytics
  - No data searching and exploring (unlike Kibana)
- Grafana dashboards are used
  - Prometheus has limited dashboarding capabilities of its own
- Prometheus and the Grafana Dashboards are installed ICP using Helm charts

Monitoring IBM Cloud Private 15 © Copyright IBM Corporation 2018

IBM Training

Logging Components

- Elasticsearch stack * - https://www.elastic.co/elk-stack
- Log Management system which is composed of:
  - Elasticsearch
    - Distributed, JSON-based search and analytics engine designed for horizontal scalability, maximum reliability, and easy management
  - Logstash
    - Dynamic data collection pipeline with an extensible plugin ecosystem and strong Elasticsearch synergy
  - Kibana
    - Extensible user interface for configuring and managing all aspects of the Elastic Stack
    - Comprehensive log analytics dashboard to visualize and build dashboards for log data
  - Beats
    - Platform for lightweight shippers that send data from edge machines to Logstash and Elasticsearch
- Packaged as a series of Helm charts
  - Deployed automatically when ICP is installed
  - Can be customized and/or opted-out of

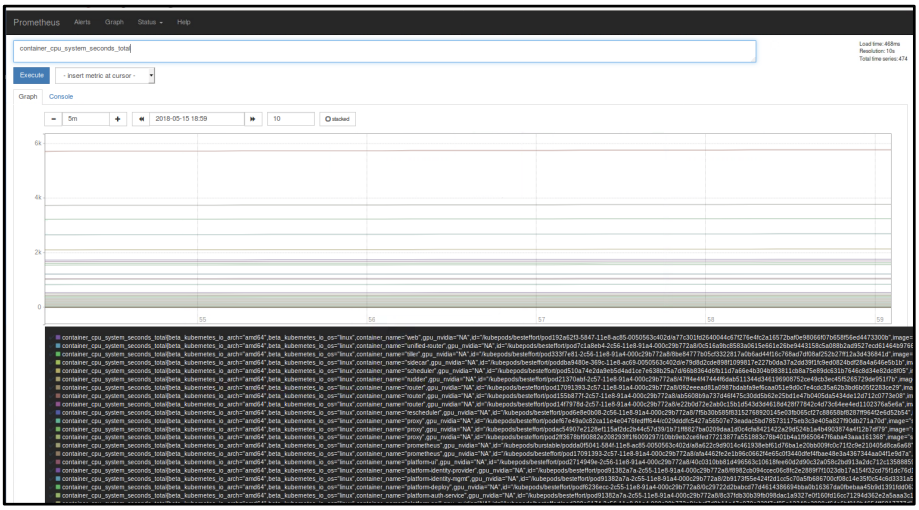
* Previously known as ELK Stack

Monitoring IBM Cloud Private 16 © Copyright IBM Corporation 2018

IBM Training

Monitoring an IBM Cloud Private Deployment

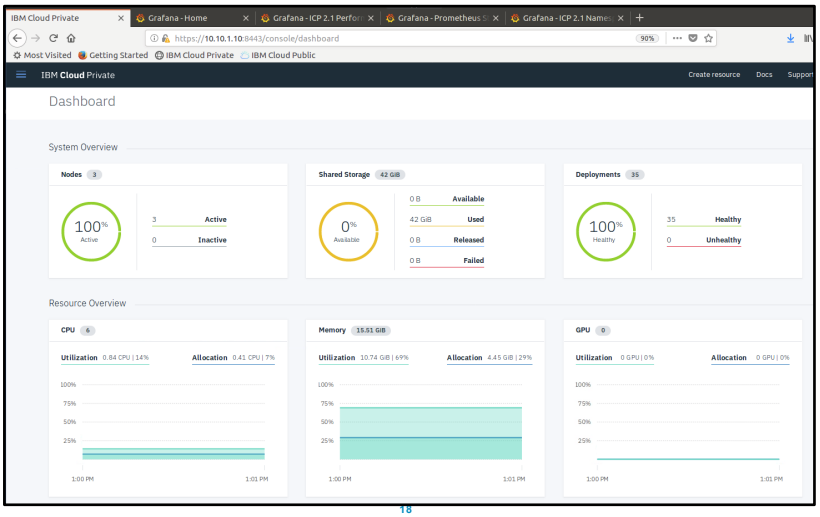
- ICP Prometheus display
  - Data is there but hard to absorb



IBM Training

Monitoring an IBM Cloud Private Deployment

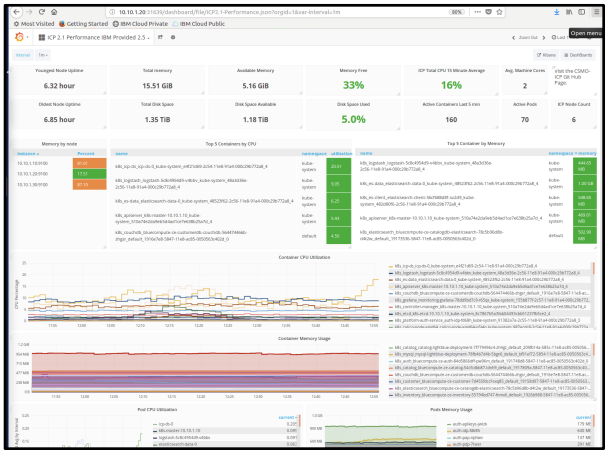
- ICP dashboard shows a high level overview of resource usage and status



IBM Training

Monitoring an IBM Cloud Private Deployment

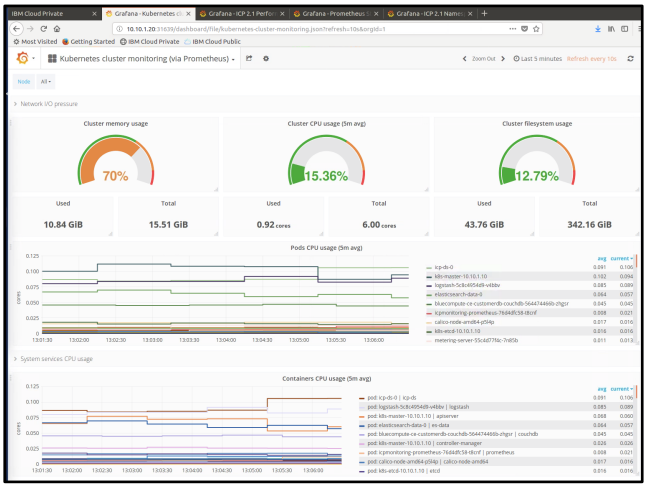
- ICP-provided Grafana dashboard shows more detail of resource usage and status



IBM Training

Monitoring an IBM Cloud Private Deployment

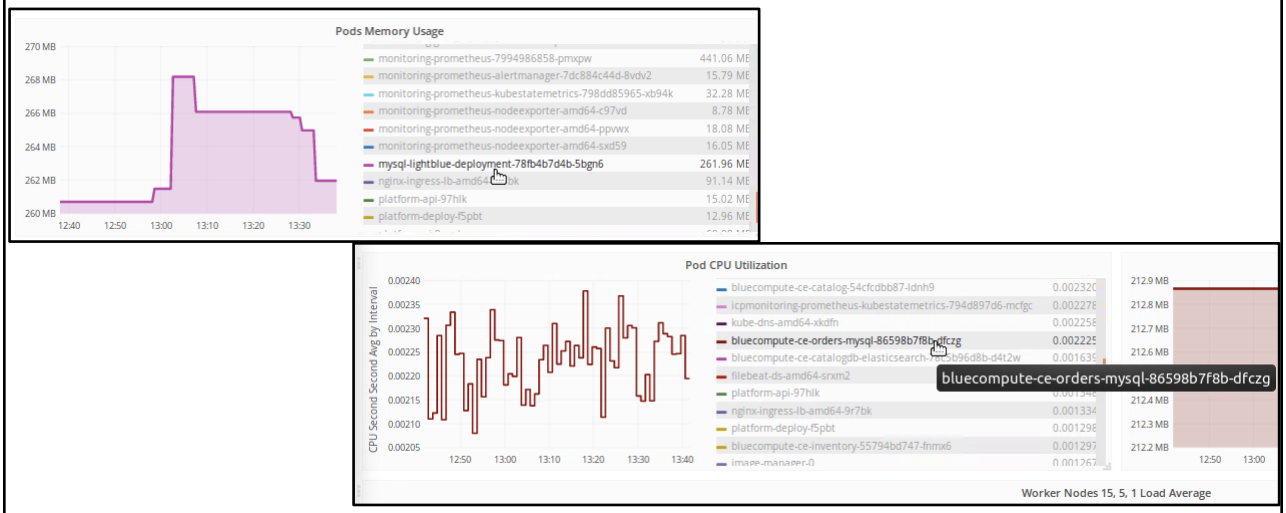
- Kubernetes Dashboard shows a variety of cluster metrics



IBM Training

Monitoring an IBM Cloud Private Deployment

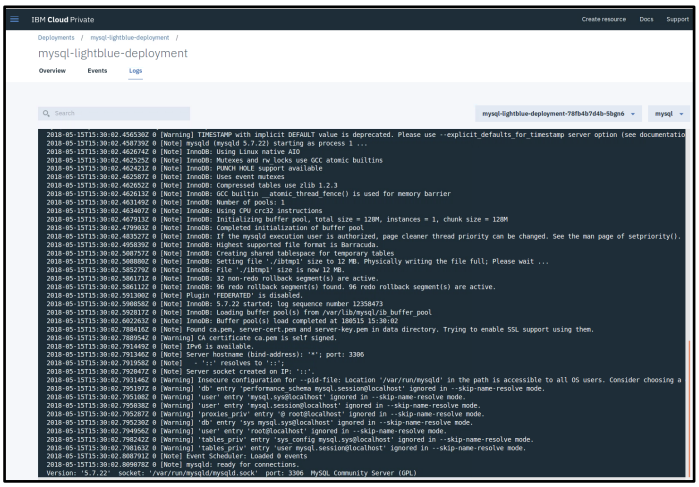
- Within an application you can view metrics of the pod



IBM Training

Monitoring an IBM Cloud Private Deployment - Logs

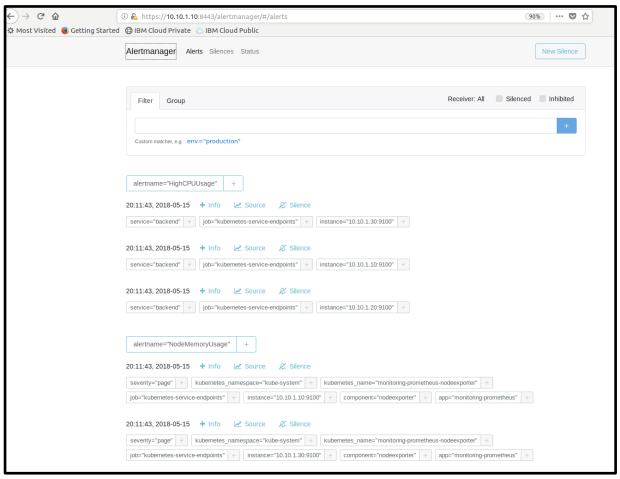
- Logging is also available for more detailed debugging



IBM Training

Monitoring an IBM Cloud Private Deployment - Alerts

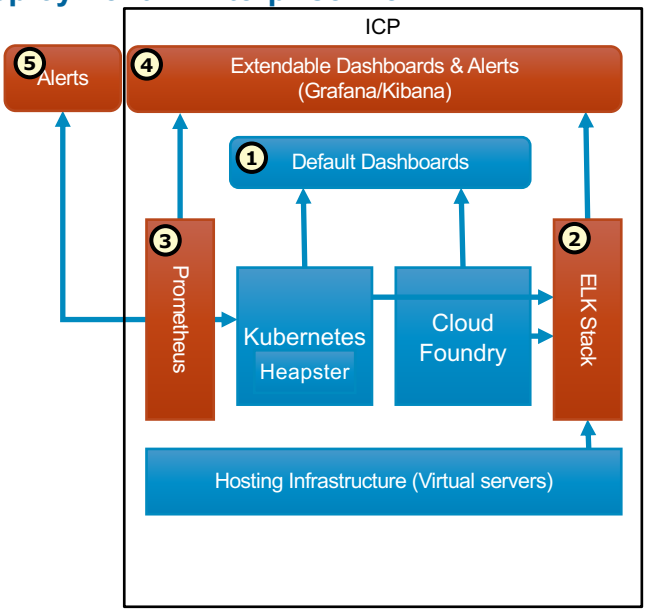
- Organizations want to monitor and alert on Out of Bounds conditions for the applications and infrastructure



IBM Training

Managing an IBM Cloud Private Deployment – Enterprise View

- Enterprises want to consolidate operational and monitoring resources
  - Prometheus can send alerts to external enterprise management system such as IBM Application Performance Management (APM)
- Monitoring and logging part of some enterprises larger operational objective …



IBM Training

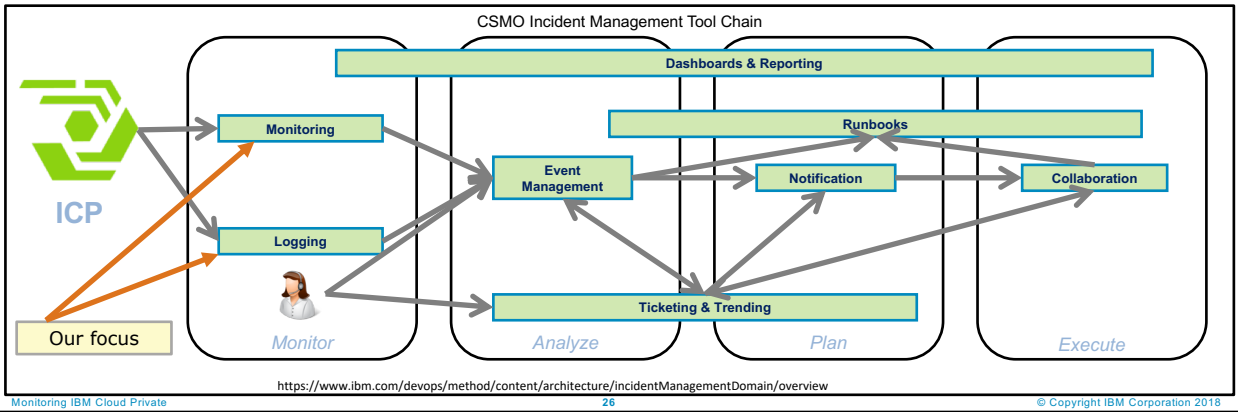
Cloud Service Management and Operations

- What is it?
  - All the activities an organization does to plan, design, deliver, operate, & control the IT & cloud services it offers
  - Includes the operational aspects of applications and services
    - Once pushed to production, an application must be managed and monitored to ensure availability and performance according to service level agreements (SLAs) or service level objectives (SLOs)
- Why is it important?
  - Poor application performance will drive customers away
  - As organizations move workloads to cloud faster and faster it's important to know how the workload is performing against customer demand
  - The ability to rapidly respond and correct anomalies is impossible without:
    - Proper monitoring of workload and associated infrastructure
    - Incident notification and management
  - Performance Management data provides the ability to trend the application performance and understand it growth and seasonal patterns

IBM Training

CSMO Incident Management Tool Chain

- Incident management
  - The practice of restoring a damaged service to health as quickly as possible
- Incident management tool chain
  - Set of tools used to aid incidence management
- Different levels based on enterprise need
- First level is monitoring and logging



IBM Training

Unit Summary

- Monitoring approaches in IBM Cloud Private
- What is available out of the box
- Cloud Service Management and Operations

IBM Training

Microservices Application Architecture

IBM Training

Prerequisites

- Developing cloud applications
  - Cloud-native application development
  - Twelve-Factor Apps
- Deploying cloud applications
  - Docker and Kubernetes in IBM Cloud Container Service
  - Cloud Foundry in IBM Cloud
  - Swarm in Docker Enterprise Edition for IBM Cloud
  - OpenWhisk in IBM Cloud Functions
- IBM Cloud capabilities
  - IBM Cloud architecture
  - WebSphere Liberty for microservices using Java, Java EE, and MicroProfile

IBM Training

Unit objectives

- List the benefits of cloud native applications and the microservices application architecture
- Describe how the microservices application architecture compares with existing application architectures
- Show how an application can be designed as a set of microservices

IBM Training

Lesson 1
Introduction to microservices

IBM Training

The curse of the unmaintainable application

Back in 2005, the applications for each of Expedia's travel sites were too difficult to modify

"Each site was an interconnected tangle of code. If programmers wanted to add the ability to handle credit card billing in Germany, for example, that might end up causing errors in the flight-search feature. As a result, changes were rare: A site would be updated only a couple times a year."

Don't let this happen to you (any more)!

"Getting to Know You: Expedia has bet everything on understanding the psyche of the modern traveler" in *Bloomberg Business Week*, February 25, 2016
http://www.bloomberg.com/news/features/2016-02-25/expedia-thinks-it-can-help-you-find-the-dream-vacation-you-didn-t-know-you-wanted

IBM Training

Cloud native

An ***application architecture*** designed to use the *strengths* and accommodate the *challenges* of a standardized cloud environment, including:

- Elastic scaling
- Immutable deployment
- Disposable instances
- Less predictable infrastructure

IBM Training

What it means to be cloud native

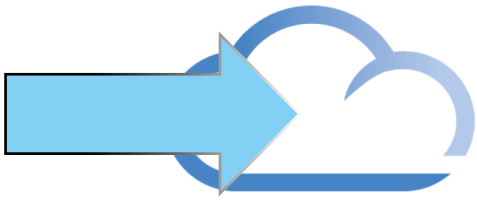
- Clean contract with underlying OS to ensure maximum portability
- Scale elastically without significant changes to tooling, architecture, or development practices
- Resilient to inevitable failures in the infrastructure and application
- Instrumented to provide both technical and business insight
- Use cloud services such as storage, queuing, and caching
- Rapid and repeatable deployments to maximize agility
- Automated setup to minimize time and cost for new developers

IBM Training

Microservices

An **engineering approach** focused on **decomposing** an application into **single-function** modules with **well defined interfaces** which are **independently** deployed and operated by **small teams** who own the **entire lifecycle** of the service

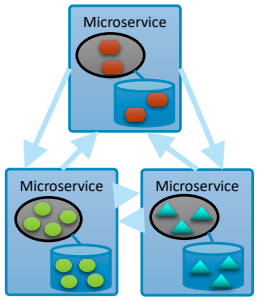
Microservices accelerate delivery by **minimizing communication** and coordination between people while **reducing the scope** and risk of change



IBM Training

Microservices: Using technology more efficiently

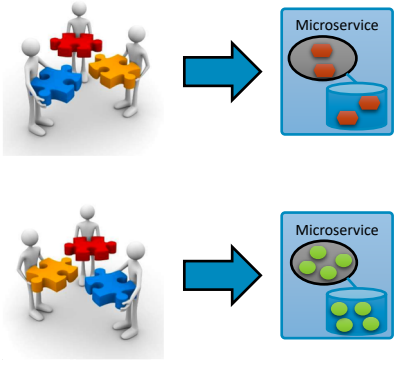
- Microservices is an application architectural style
  - The application is composed of microservice components
- Microservice, a component in this architecture
  - Each is a miniature application
  - Each is focused on one task, a business capability
    - The Single Responsibility principle
  - Each can be deployed and updated independently
  - They are loosely coupled
  - Each has a well-defined interface
    - REST APIs



IBM Training

Microservices: Making developers more efficient

- Each microservice is developed and deployed by a small team
  - The team owns the entire lifecycle of the service
- Microservices accelerate delivery
  - Minimize communication and coordination between people
    - "Less time in meetings, more time coding"
  - Reducing the scope and risk of change
- Microservices facilitate agile development
  - They make their development teams loosely coupled



IBM Training

Microservices are an evolution

Evolution of architectural styles

- **Monolithic**
  One large application that does everything
- **Microservices**
  Several smaller applications that each does part of the whole

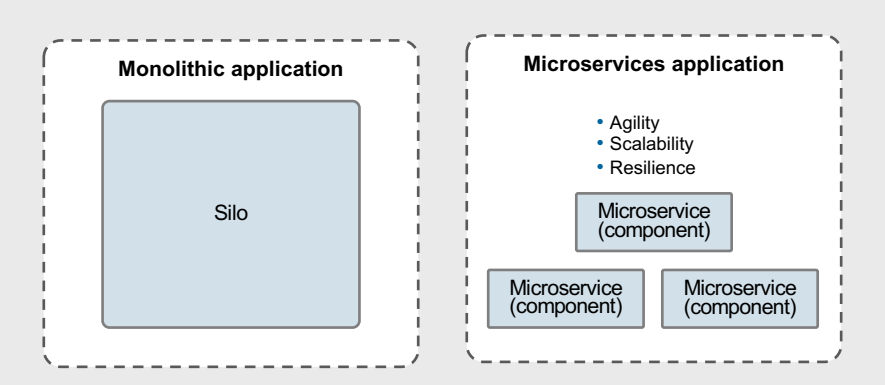
Evolution of service orientation

- **SOA**
  Focused on reuse, technical integration issues, technical APIs
- **Microservices**
  Focused on functional decomposition, business capabilities, business APIs

IBM Training

Microservices architecture

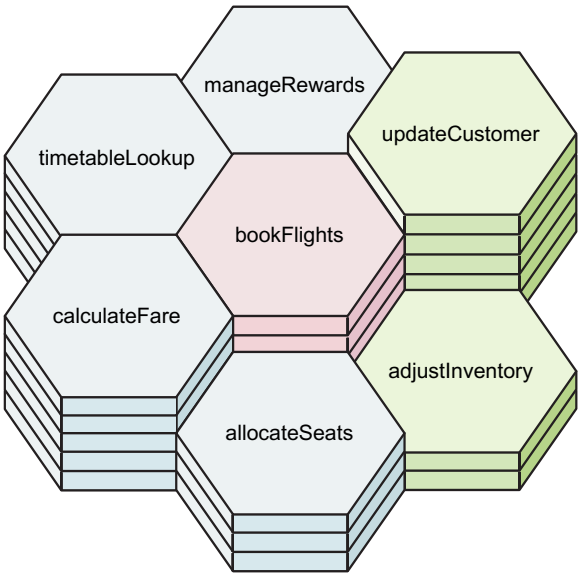
- Microservices architecture is about breaking down large silo applications into more manageable, fully decoupled pieces
- A *microservice* is a granular, decoupled component within a broader application



IBM Training

Sample application that uses microservices

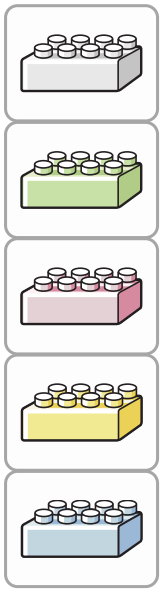
- Airline reservation application
  - Seven services (in this example)
- Each service includes
  - Logging
  - Metrics
  - Health check
  - Service endpoint
  - Service registry
  - Service management
- What do the different colors of the tiles mean (red, blue, and green)?
- The tiles are in stacks, some higher than others. What does the height mean?



IBM Training

Key tenets of a microservices architecture

1. Large monoliths are broken down into many small services
   - Each service runs in its own process
   - One service per container
2. Services are optimized for a single function
   - There is only one business function per service
   - The Single Responsibility Principle
     - A microservice should have one, and only one, reason to change
3. Communication via REST API and message brokers
   - Avoid tight coupling introduced by communication through a database
4. Per-service continuous integration and continuous deployment (CI/CD)
   - Services evolve at different rates
   - You let the system evolve but set architectural principles to guide that evolution
5. Per-service high availability (HA) and clustering decisions
   - One size or scaling policy is not appropriate for all
   - Not all services need to scale; others require autoscaling up to large numbers



IBM Training

Advantages of microservices

- Developed independently
  - Limited, explicit dependencies on other services
- Developed by a single team
  - The team is small
  - All team members can understand the entire code base
- Developed on its own timetable
  - New versions delivered independently of other services
- Polyglot: Each can be developed in a different language
  - Select the best language
- Manages its own data
  - Select the best technology and schema
- Scales and fails independently
  - Isolates problems

IBM Training

Comparing monolithic and microservices architectures

Category	Monolithic architecture	Microservices architecture
Architecture	Built as a single logical executable	Built as a suite of small services
Modularity	Based on language features	Based on business capabilities
Agility	Changes require building and deploying a new version of the entire application	Changes can be applied to each service independently
Scaling	Entire application scaled when only one part is the bottleneck	Each service scaled independently when needed
Implementation	Typically entirely developed in one programming language	Each service can be developed in a different programming language
Maintainability	Large code base is intimidating to new developers	Smaller code bases easier to manage
Deployment	Complex deployments with maintenance windows and scheduled downtimes	Simple deployment as each service can be deployed individually, with minimal downtime

IBM Training

Technology advances have made microservices possible

- Ease and feasibility of distributing components
  - Internet, intranet, or network maturity
  - RESTful API conventions or *perceived* simplicity, and lightweight messaging
- Ease and simplicity of hosting
  - Lightweight runtimes
    - Examples: Node.js and WebSphere Liberty
  - Simplified infrastructure
    - OS virtualization (hypervisors), containerization (Docker), infrastructure as a service (cloud infrastructure)
    - Workload virtualization (examples: Cloud Foundry, Kubernetes, OpenWhisk, Swarm)
  - Platform as a service
    - Autoscaling, SLA management, messaging, caching, build management
- Agile development methods
  - Examples: IBM Cloud Garage Method, XP, TDD, Scrum, Continuous Delivery
  - Standardized code management, such as GitHub

IBM Training

Microservice challenges

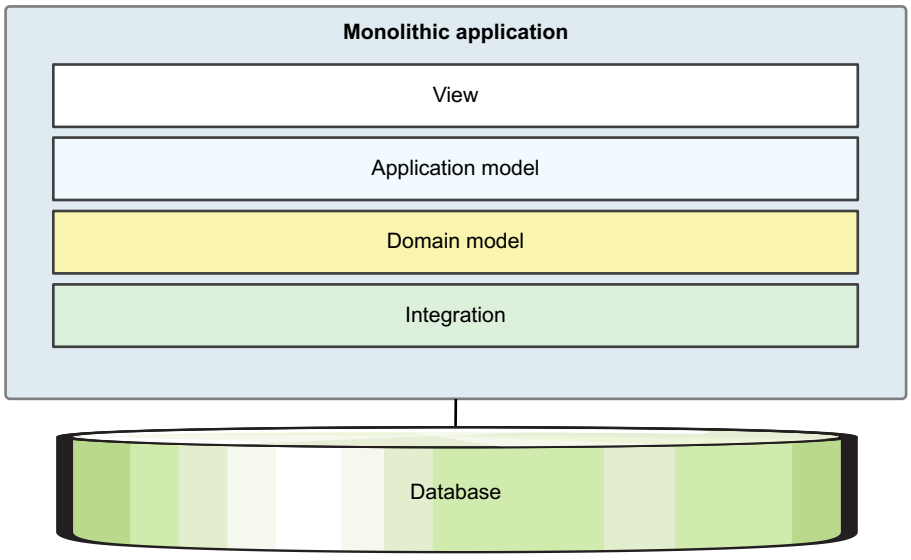
- Greater operational complexity because there are more moving parts to monitor and manage
- Developers must have significant operational skills (DevOps)
- Service interfaces and versions
- Duplication of effort across service implementations
- Extra complexity of creating a distributed system
  - Network latency
  - Fault tolerance
  - Serialization
- Designing decoupled, non-transactional systems is difficult
- Locating service instances
- Maintaining availability and consistency with partitioned data
- End-to-end testing

IBM Training

Lesson 2
Application architecture evolution

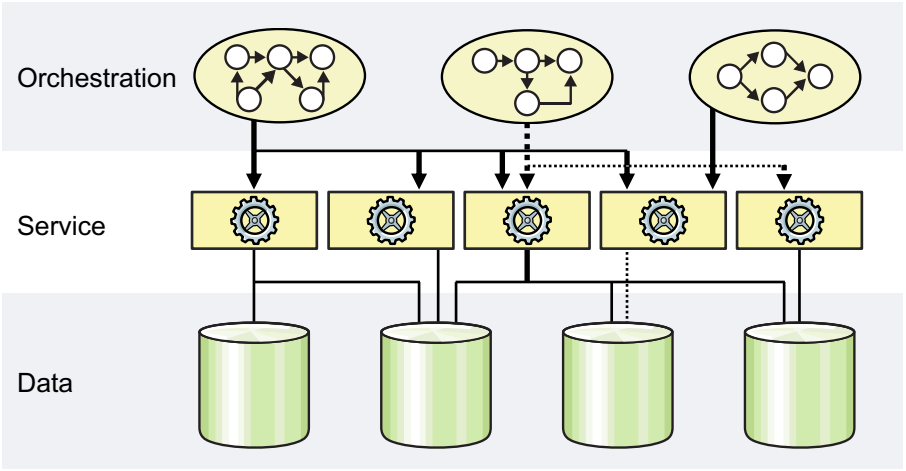
IBM Training

Layered application architecture



IBM Training

SOA stack



IBM Training

Microservices and SOA

Both SOA and microservices deal with a system of services that communicate over a network

But there are differences

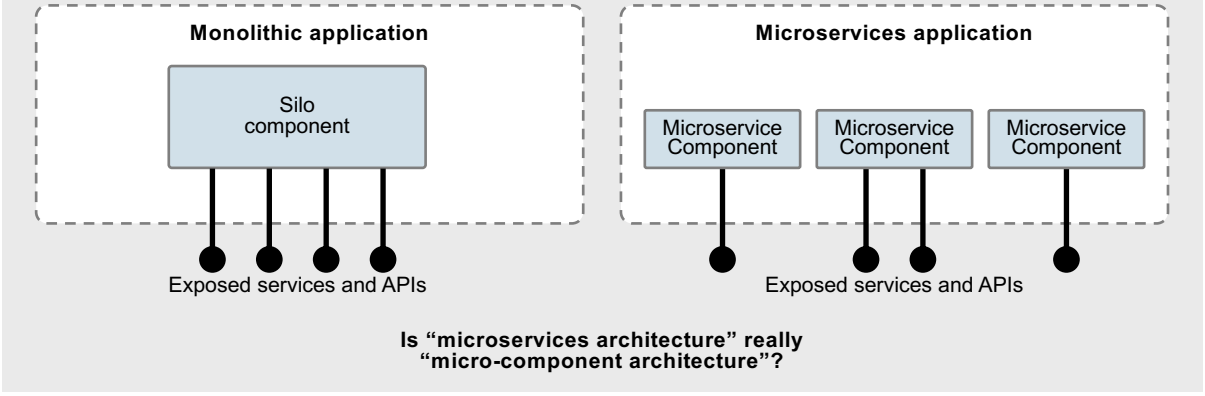
- The focus of SOA is on reuse
  - This tends to align with a centrally funded model
  - SOA services tend to be “servants of many masters”
  - This means that a change to a SOA service might impact multiple consumers
- The focus of microservices is on breaking down a potentially monolithic application into smaller, more manageable components
  - With the objective of more flexible, decoupled, faster development
  - Challenges here relate to needing good DevOps, management views, and controls

IBM Training

Common misconception that results from the term "microservice"

~~Microservices are more fine-grained web services~~
~~APIs are microservices~~

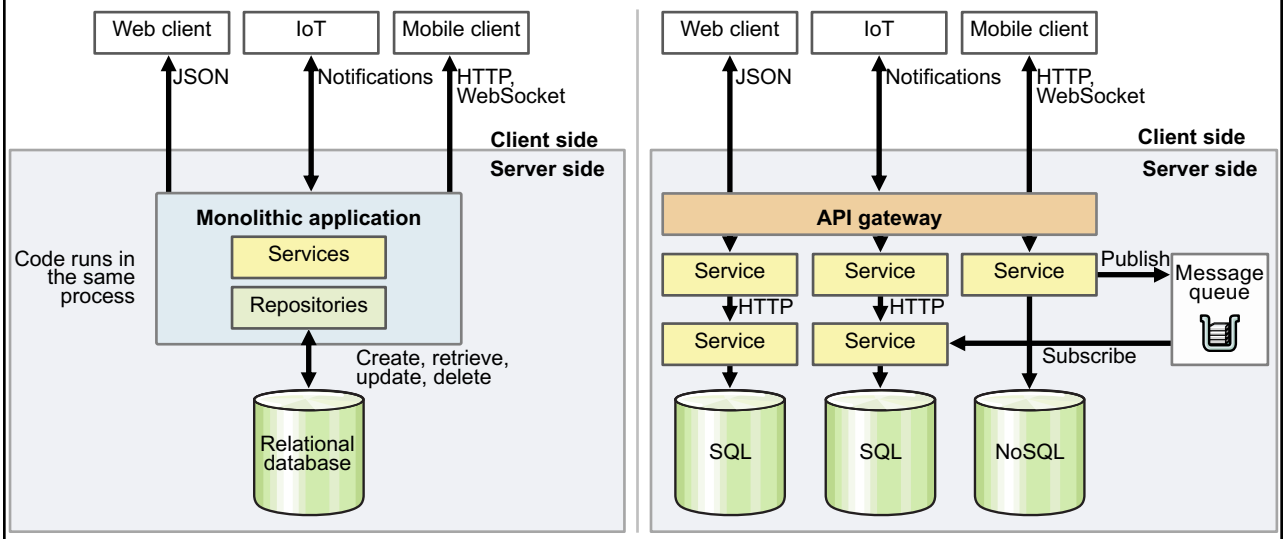
"Micro" refers to the granularity of the ***components***,
not the granularity of the exposed interfaces



Is "microservices architecture" really "micro-component architecture"?

IBM Training

Monolithic architecture versus microservices architecture

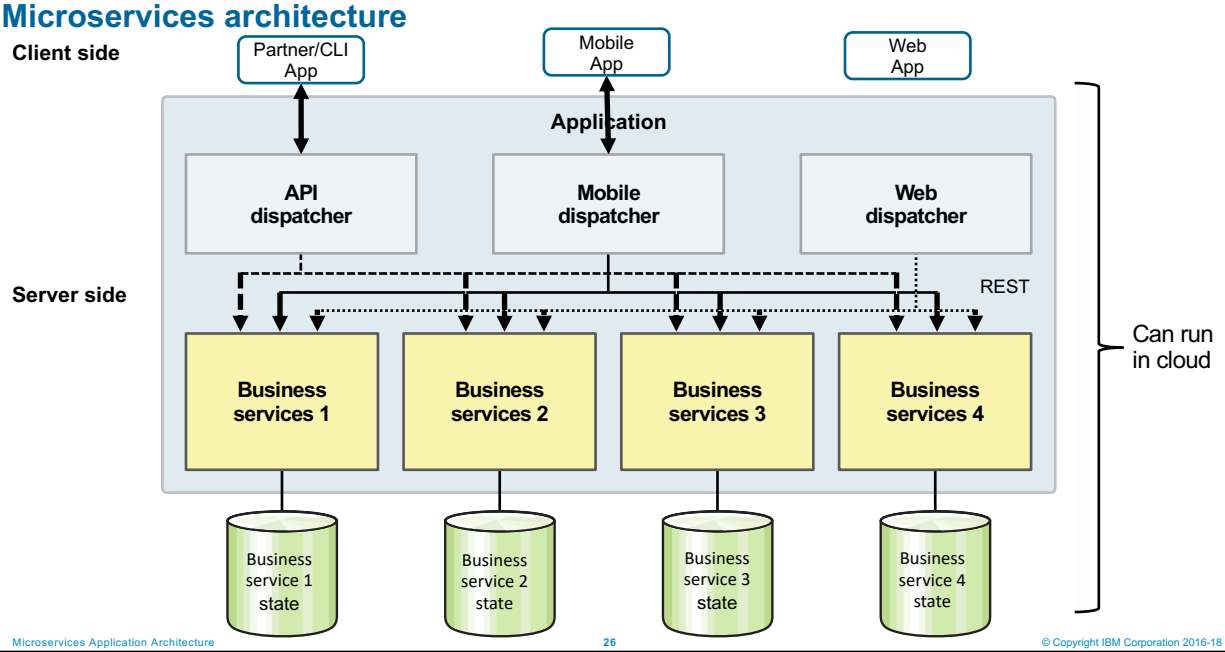


IBM Training

Lesson 3
Microservices component architecture

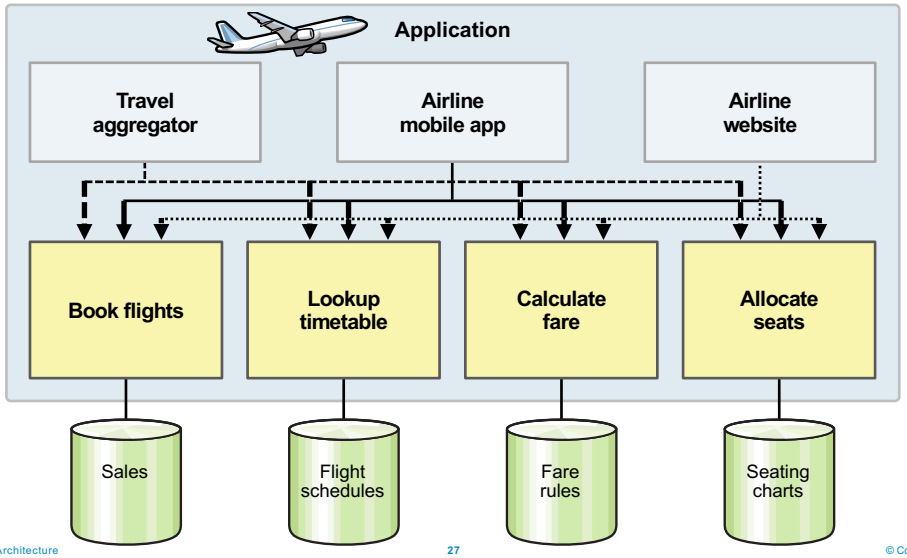
IBM Training

Microservices architecture



IBM Training

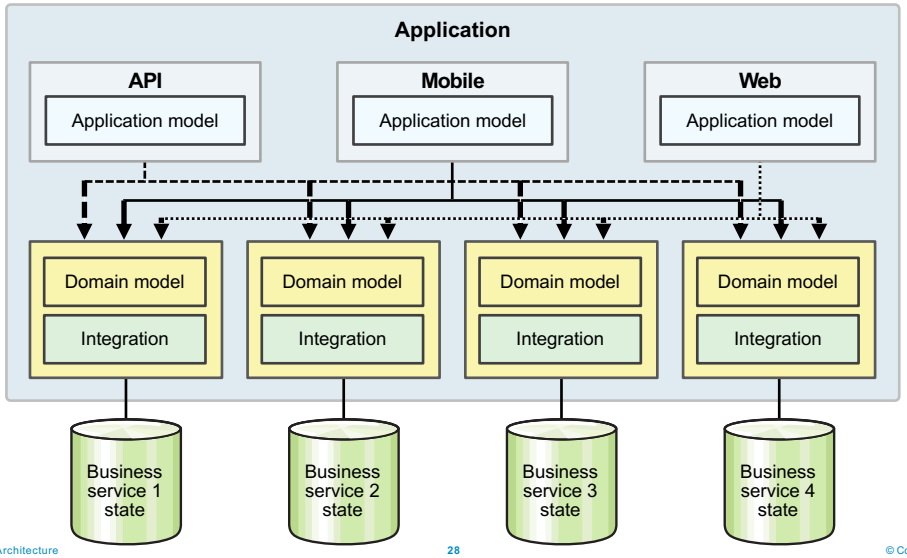
Sample airline architecture



IBM Training

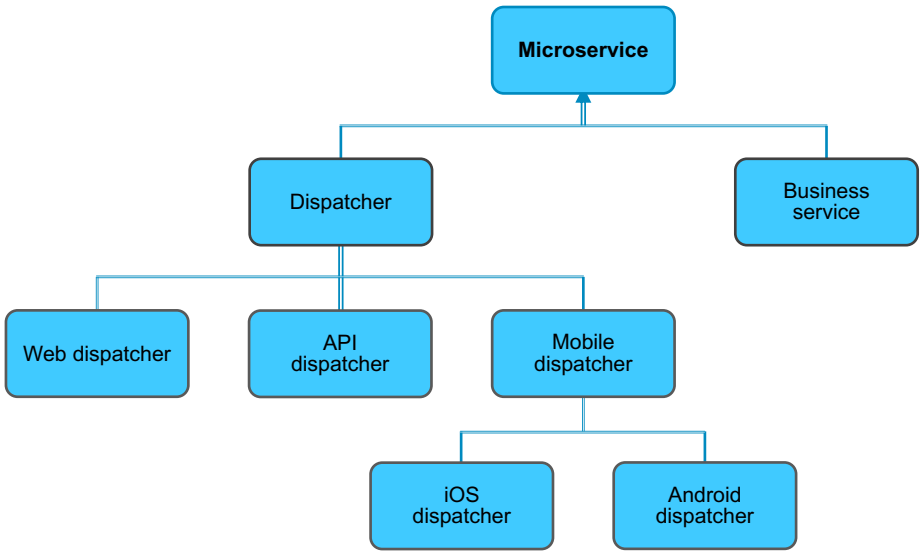
Microservices layers

Each service is a smaller half-monolith



IBM Training

Microservices types hierarchy



IBM Training

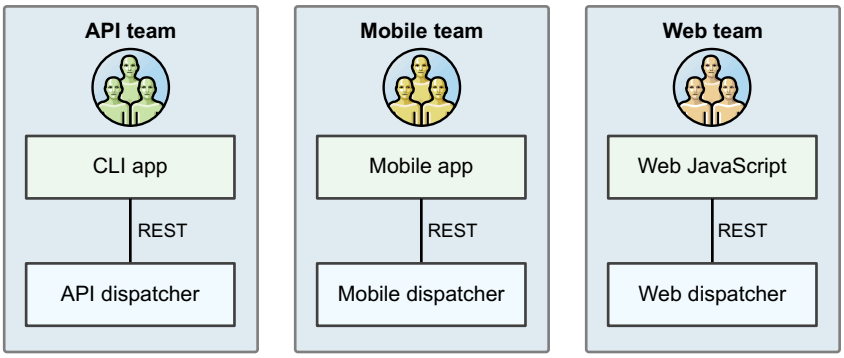
Language decisions

- Dispatchers are most often written in Node.js
  - Better fidelity with the clients
    - Especially clients that run JavaScript
    - iOS teams might want to use the Swift server-side runtime
  - I/O intensive, supports numerous concurrent clients
- Business services are most often written in Java
  - Better for CPU-intensive tasks
  - Better connectivity to external systems
- The team selects the language
  - Choose the language that best fits the job

IBM Training

Back ends for front ends

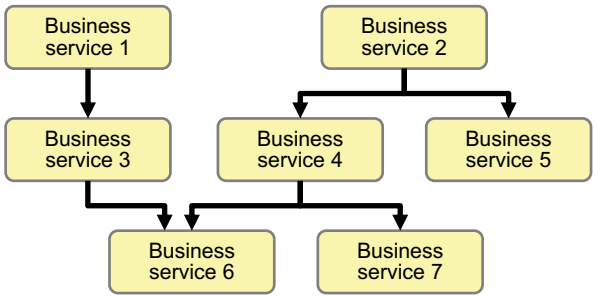
- Each dispatcher is a back end for an external front end, typically a UI
- The same team develops each back end and front end pair
  - Use the same or compatible languages in the pair



IBM Training

Business service microservices dependencies: Typical

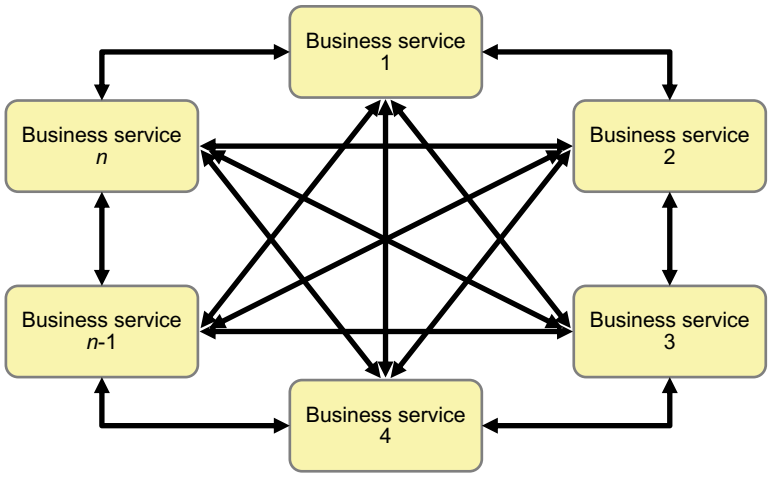
- Business services can delegate to other business services
  - Try to avoid circular dependencies
- Be careful that each service still implements a complete task
  - They are not separate layers



IBM Training

Business service microservices dependencies: Death Star

- Practically every service delegates to or coordinates with every other service
- Difficult to deploy and maintain



IBM Training

Conclusion

IBM Training

Unit summary

- List the benefits cloud native applications and the microservices application architecture
- Describe how the microservices application architecture compares with existing application architectures
- Show how an application can be designed as a set of microservices

IBM Training

Next

- More on microservices
  - Microservice development
  - Service mesh
- Deploying cloud applications
  - Docker and Kubernetes in IBM Cloud Container Service
  - Cloud Foundry in IBM Cloud
  - Swarm in Docker Enterprise Edition for IBM Cloud
  - OpenWhisk in IBM Cloud Functions
- IBM Cloud capabilities
  - IBM Cloud architecture
  - WebSphere Liberty for microservices using Java, Java EE, and MicroProfile
  - Cloud data services
  - Integration with systems of record
- DevOps
  - Continuous delivery of microservices

IBM Training

Resources

- *Microservices from Theory to Practice: Creating Applications in IBM Bluemix Using the Microservices Approach*, IBM Redbook, 26 August 2015, http://bit.ly/ibmredmicro
- "Microservices, SOA, and APIs: Friends or enemies?" by Kim J. Clark, *IBM developerWorks*, 21 January 2016
- "Microservices.TV" https://developer.ibm.com/tv/microservices/
- "Microservices in action, Part 1: Introduction to microservices" by Rick Osowski, *IBM developerWorks*, 26 June 2015
- "Introduction to Microservices and Cloud Native Application Architecture" by David Currie
- "Microservices" by James Lewis and Martin Fowler, http://bit.ly/microsvc
- *Building Microservices: Designing Fine-Grained Systems* by Sam Newman, O'Reilly Media, February 2015, http://bit.ly/oreillymicro
- "Netflix Hystrix: How It Works" https://github.com/Netflix/Hystrix/
- Spring Cloud https://cloud.spring.io
- *Release It!: Design and Deploy Production-Ready Software* by Michael T. Nygard, Pragmatic Bookshelf, April 2007
- *Domain-Driven Design: Tackling Complexity in the Heart of Software* by Eric Evans, Addison-Wesley Professional, August 2003
- *Enterprise Integration Patterns* by Gregor Hohpe and Bobby Woolf, Addison-Wesley Professional, October 2003

IBM Training

IBM Cloud Data Services Overview

IBM Training

Topics

- Where and how will data be used?
- Why use NoSQL?
- Attributes of NoSQL databases
- Implementation: Cloudant

IBM Training

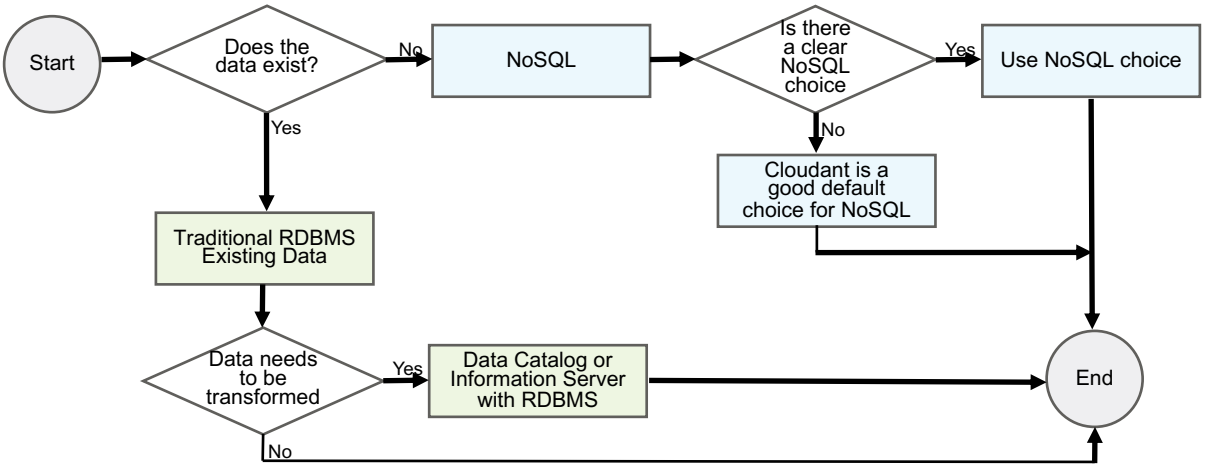
Polyglot Persistence

- A basic assumption is that your application will probably have to store some data persistently
  - For longer than the length of a single user session
- At one time, it was a simple assumption that Web programs would run on a relational database
  - Multiple relational database options exist for IBM Cloud Public
- The assumption of using a SQL database is challenged by the data management options collectively called “NoSQL”
- These data stores are grouped into four basic types
  - Key-value
  - Document
  - Column-family
  - Graph

IBM Cloud Data Services Overview 3

IBM Training

What data service should be used?



IBM Cloud Data Services Overview 4

IBM Training

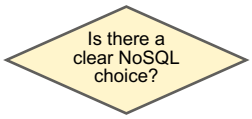
Ask the right questions about the data

- Is there existing data in a relational database management system (RDBMS)?
- For cloud native development with new data don't create a new RDBMS
  - Initially it might seem to be easier and quicker
  - Can hinder rapid development
  - Traditional RDBMS requires structure, a schema
- What does the data "look like"
- Existing data probably already lives in an RDBMS
  - That doesn't mean you need to use it "as is"
  - There are services for data transformation and caching
- NoSQL databases provide:
  - Faster prototyping
  - No need for strict data typing, schema-less
  - No need to throw away non-conforming data

IBM Training

Choosing NoSQL over RDBMS is not the final answer

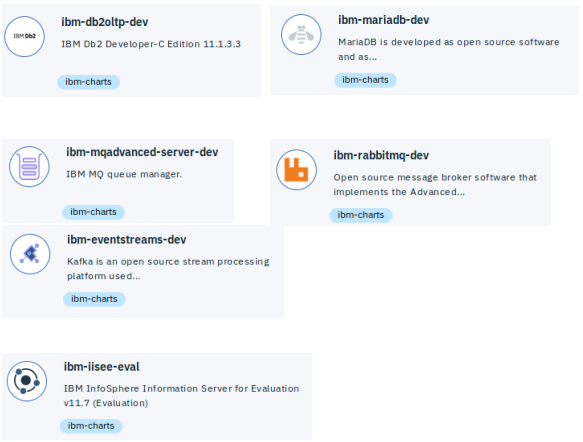
- Look to see what type of problem you're trying to solve. Let the structure of your data and your queries define your technology choice.
- Simple key-value cache?
  - Use Redis
- Storing documents?
  - Use Cloudant
- Representing graph data?
  - Use Compose for JanusGraph
- Cloudant is a good default NoSQL database



IBM Training

Options to work with SQL data in Cloud

- Use a provider with SQL data in cloud:
  - Compose databases
  - Db2, mysql, postgres, etc
  - Run database container with persistent volume
- Connect to data source outside the cloud
  - MQ
  - Message Hub
  - Kafka
- Transform and cleanse data to cloud
  - InfoSphere Information Server



IBM Training

Topics

- Where and how will data be used?
- Why use NoSQL?
- Attributes of NoSQL databases
- Implementation: Cloudant

IBM Training

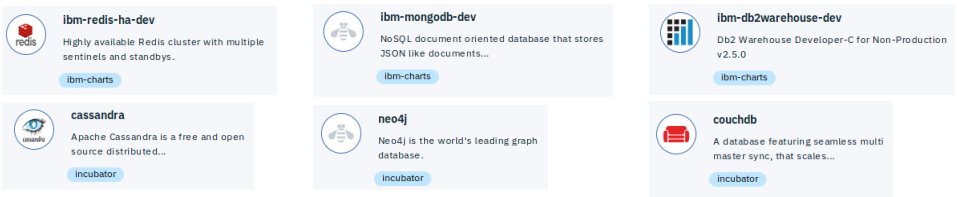
Why use NoSQL databases

- Some data does not need to be normalized or categorized
  - CLOB, BLOB, reports, customer reviews, and so on
- If the data would be stored as a single unit, it is a waste of effort to write it to columns and tables of an RDBMS
  - A NoSQL database can handle this type of data much more rapidly
- Horizontal scaling
  - Query massive amount of data
  - Parallel processing is a common capability
- Designed to be reliable on unreliable hardware
  - NoSQL databases are distributed

IBM Training

NoSQL Landscape

Database Type	Examples	When best used
Key-Value	Redis, Memcached, DynamoDB	Storing interaction data, user preferences, simple shopping carts. Not great at set operations.
Document	Cloudant, MongoDB	Event logging, content management, analytics. Not great at complex queries.
Column-Family	Cassandra, Db2 Warehouse	Event logging, counters, blogs. Not compatible with ACID transactions.
Graph	GraphDB, JanusGraph, Neo4J, OrientDB	Social Networks, Location-based Services. Not optimal for certain types of bulk updates.



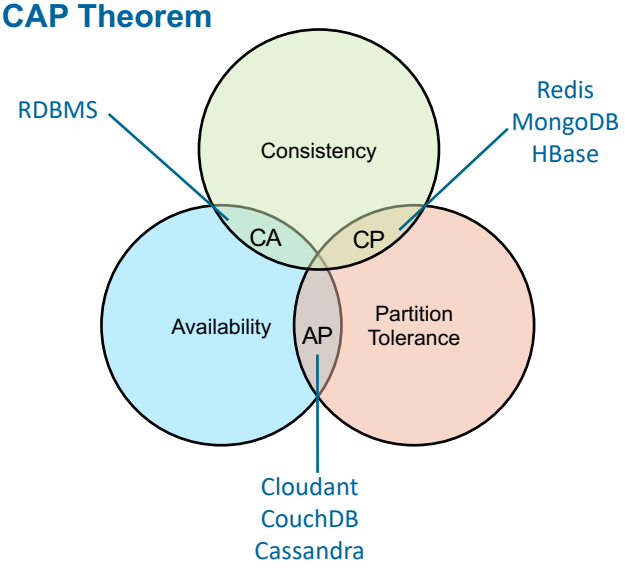
IBM Training

Topics

- Where and how will data be used?
- Why use NoSQL?
- Attributes of NoSQL databases
- Implementation: Cloudant

IBM Training

CAP Theorem



CAP Theorem says you can only two:
- Consistency
- Availability
- Partition Tolerance

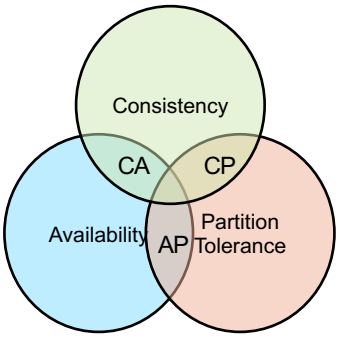
Horizontal replicas can become partitioned
- Consequence: Choose always consistent or always available (can't have both!)
- Configurable in many NoSQL databases

Eventually consistent
- To maintain availability, partitioned replicas lose synchronization

IBM Training

NoSQL and eventual consistency

- Eventual consistency
  - Many NoSQL databases use replicas (horizontal scaling) to maintain high availability
  - The network between database replicas can become partitioned
  - CAP Theorem: When partitioned, either consistency or availability must be sacrificed
  - To maintain availability, most databases are configured to sacrifice consistency
  - When the partitioning is resolved, synchronization reestablishes consistency
- Consequences
  - Transactions are not ACID across replicas
  - Updates are inconsistent until they replicate (usually milliseconds)
  - Applications must tolerate the possibility that two reads may produce inconsistent results
- Be aware of the limitations of your particular database choice
  - Don't assume that your data is always consistent!



IBM Cloud Data Services Overview 13

IBM Training

Eventual consistency works well

- Eventual consistency sounds problematic
  - Many business domains are eventually consistent (hotels, airlines, and so on)
- How is conflict resolution handled?
  - "Last writer wins" can overwrite data, resolution is non trivial
  - Reconciliation can be achieved by time stamps (epochs, vector clocks)
- CouchDB (Cloudant) uses Multi-Version Concurrency Control (MVCC)
  - No locking required
  - Each read request sees the most recent snapshot of the database
- Database detects update conflicts
  - Application or user can reconcile the conflicts

IBM Cloud Data Services Overview 14

IBM Training

Understanding NoSQL data retrieval

- There are several approaches for accessing data
  - These are not necessarily specific to NoSQL
  - Understanding these options can help in deciding which NoSQL to use
- Four common approaches
  1. SQL-style query
  2. Look-up by ID
  3. Object navigation
  4. Map-reduce

IBM Training

SQL-style queries

- If you have always used SQL queries, your natural tendency is to use them for your new cloud app

```
SELECT * FROM customers WHERE postal_code = '51922'
```

- Database considers all elements, filters out those that don't match

•Don't Do It

- Sequential queries perform poorly with most NoSQL databases
- NoSQL databases provide APIs for other methods to retrieve your data much more efficiently

IBM Training

Lookup by ID

- Do you already know the key or ID of the data you need?
  - Primary key access
  - Typical for relational databases
  - Available for most NoSQL databases

Yes

- Key Value Pair Retrieval
  - Provide the key and retrieve the values
  - A key-value database fits this scenario well

No

- This requires another approach for accessing the data
  - Object navigation
  - Map-reduce
- Many NoSQL databases provide built-in map-reduce functionality

IBM Training

Key-Value Store Databases

- Values are stored in relation to unique keys
- Values are retrieved by supplying their key

```
String key   = "location";
Object value = find(key);
```

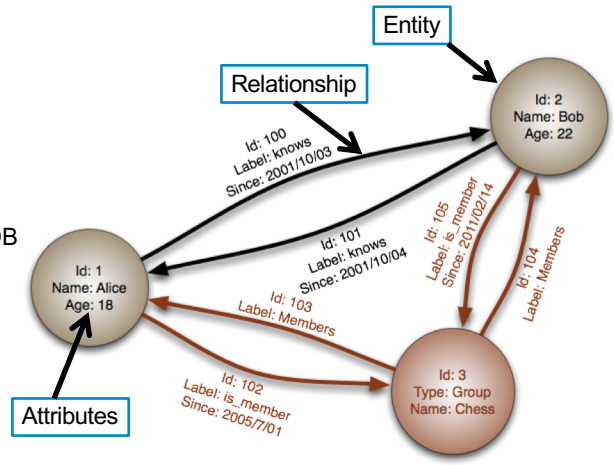
- Value data usually has no known structure
  - Any kind of data can be stored as a value
  - Very flexible
- Examples
  - **Compose for Redis**
  - **Memcached Cloud**
  - **Amazon DynamoDB**

Key	Value
firstName	Bugs
lastName	Bunny
location	Earth

IBM Training

Object Navigation

- Entity-Relationship-Attribute (ERA)
  - Each entity has attributes
  - Entities are connected by relationships
  - a.k.a. Nodes-Edges-Properties
- Implemented in graph databases
  - Examples: Compose for JanusGraph, Neo4J, OrientDB
  - Most NoSQL databases have this capability
- Object-oriented
  - Entities are objects
  - Relationships are links
  - Attributes are instance variables
- Retrieval done by following relationships
  - "Give me the members of the Chess Club"
  - Follows links rather than searching multiple tables – more efficient



IBM Training

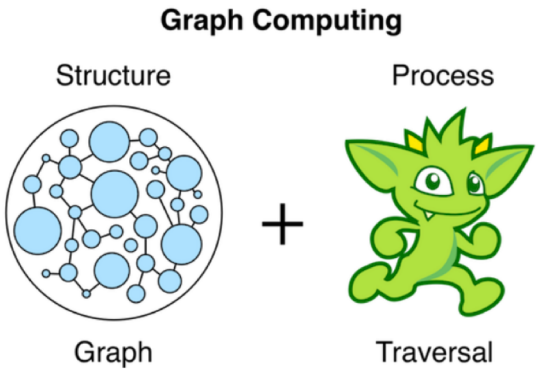
Why use a graph database?

- Relationships are important in graph databases
  - Allow more complex relationships than relational databases
  - Allow for dynamic relationships
  - Can describe relationships in greater detail
- Graph databases can be schema-less
  - Allows flexibility in data typing
- Can apply Graph Theory to databases
  - Discover disjoint sets within the data
  - Find minimal routes between nodes

IBM Training

Property Graphs

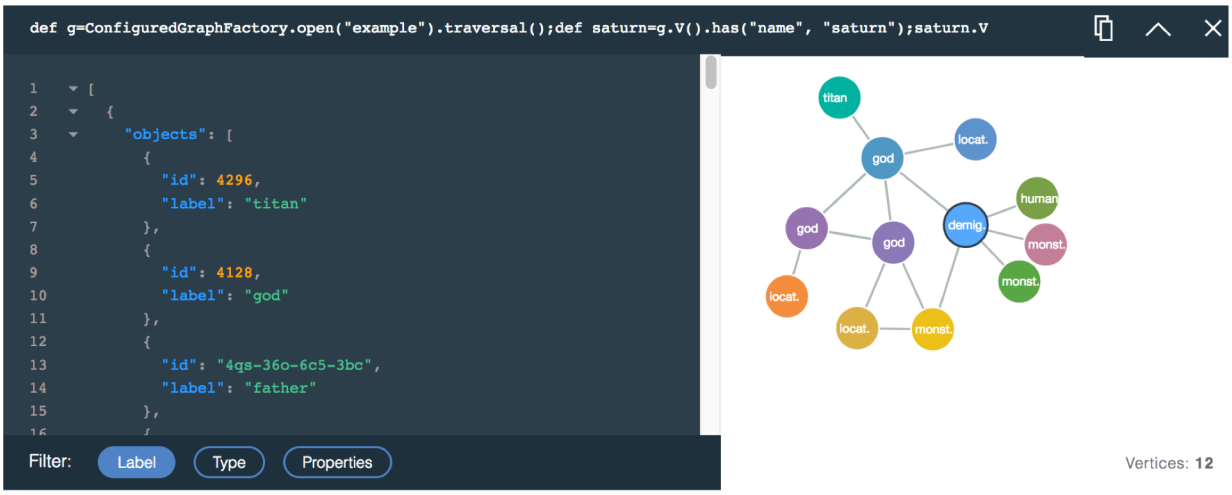
- Contains vertices and edges with properties
- Vertex represents an entity
  - A vertex has a label
  - Vertices have unique IDs
  - Vertices have properties
- Edge represents a directional relationship
  - Edges have labels
  - Each edge has a unique ID
  - Edges have properties
- Graph computing makes a distinction between the structure of the graph and the process of navigating the graph
  - JanusGraph based on TinkerPop3 and the traversal language is Gremlin



IBM Cloud Data Services Overview 21 © Copyright IBM Corporation 2016 - 2018

IBM Training

The JanusGraph data browser



IBM Cloud Data Services Overview 22 © Copyright IBM Corporation 2018

IBM Training

Map-Reduce

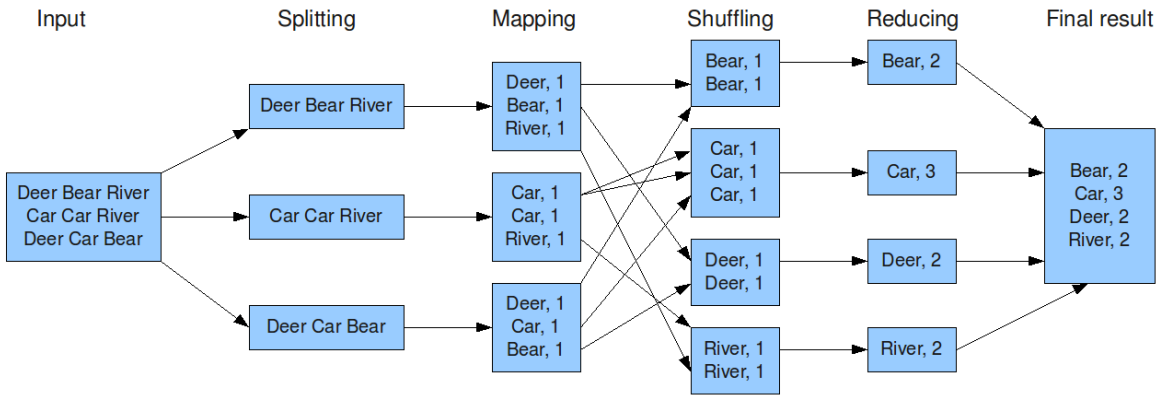
- Programming model designed to exploit parallel processing opportunities of clusters
  - Invented at Google Research
  - Horizontal scalability – job can be split across multiple servers
  - Hadoop is most well known implementation
  - Many NoSQL databases provide this capability

- Comprised of 2 major steps:

1. Map
   - Converts a set of data into intermediate key / value pairs
2. Reduce
   - Takes the output of Map step as input
   - Combines output of many map steps into a smaller set of key / value pairs

IBM Training

Consider this Example – How many times is each word used?



Note that this method could be used to answer other questions:
What word is used most, least?

IBM Training

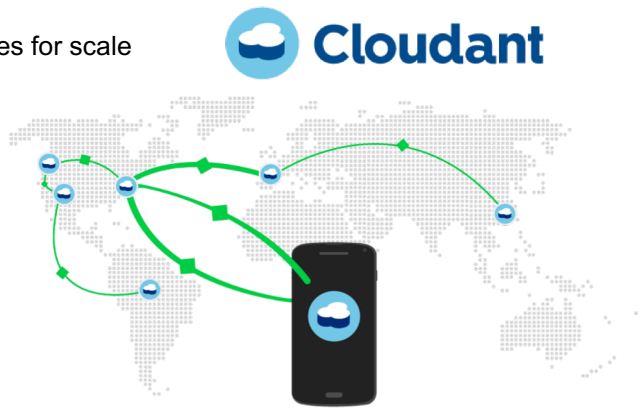
Topics

- Where and how will data be used?
- Why use NoSQL?
- Attributes of NoSQL databases
- Implementation: Cloudant

IBM Training

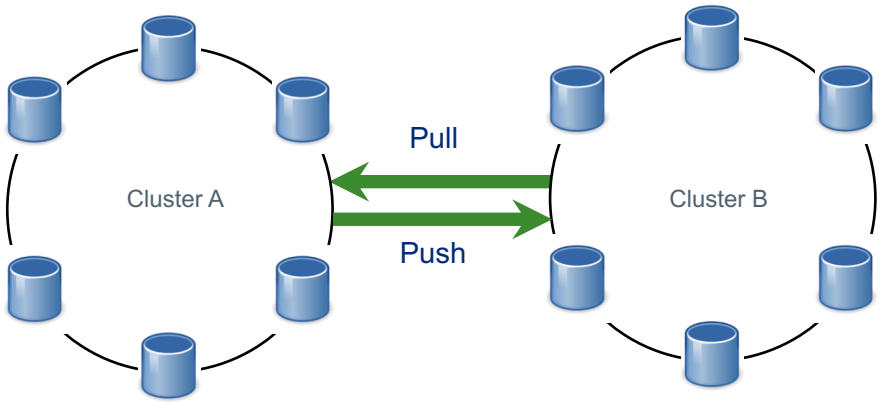
Introduction to Cloudant

- A fully managed NoSQL Database as a Service
- Transactional JSON *document* database with RESTful API
- Can spread data across data centers and devices for scale plus high availability (HA)
- Ideal for apps that require these features:
  - Massive, elastic scalability
  - High availability
  - Geolocation services
  - Full-text search
  - Occasionally connected users



IBM Training

Cloudant database replication



- Bi-directional (push/pull == sync)
- Continuous or point-in-time
- Filterable (replicate subset of docs in a database)
- Durable (replication will pick up where it left off)
- Allows for eventual consistency
- Separate clusters
  - In different regions, active-active
  - Can be used to cache data on a phone

IBM Training

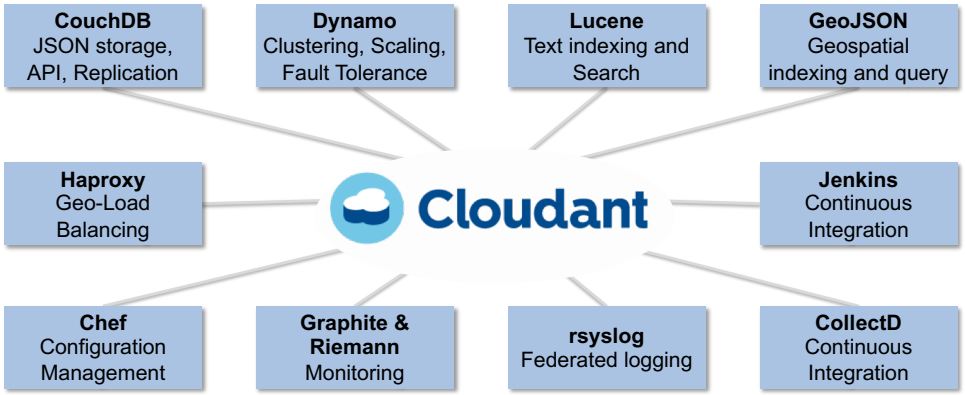
Cloudant API and Query Options

JSON Documents	Primary Index	Secondary Indices	Search	Cloudant Geospatial	Cloudant Query
• Direct doc lookup by _id	• Exists OOTB • Stored in a b-tree • Primary key → doc._id	• Built using MapReduce • Stored in a B-tree • Key → user-defined field(s)	• Built using Lucene • FTI: Any or all fields can be indexed	• Stored in R* tree • Lat/Long coorindates in GeoJSON	• Mongo-style querying • Not a 1:1 mapping • Built natively in Erlang
• Use when you want a single document and can find by the doc _id	• Use when you can find documents based on their _id • Pull back a range of keys	• Use when you need to analyze data or get a range of keys • Ex: count data fields, sum/average numeric results, advanced stats, group by date, etc.	• Ad-hoc queries • Find docs based on their contents • Can do groups, facets, and basic geo queries (bbox & sort by distance)	• Complex geometries (polygon, circularstring, etc.) • Advanced relations (intersect, overlaps, etc.)	• Ad-hoc queries • Lots of operators (>, <, IN, OR, AND, etc.) • Intuitive for people coming from Mongo or SQL backgrounds

IBM Training

Cloudant and other technologies

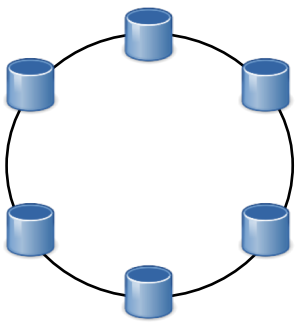
- Cloudant combines the best open source technology and thinking to create the most scalable, flexible, always-on DBaaS for big mobile and the Internet of Things (IoT)



IBM Training

Cloudant HTTP RESTful API

- CRUD operations, document retrieval using indexes, search
- Agnostic to programming language
- Apache CouchDB compatibility



GET */<database>/<doc._id>*

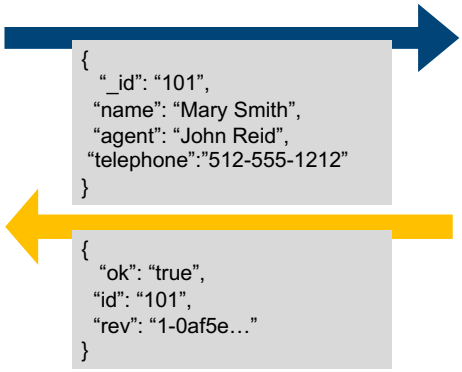
```
{
        "author" : "John Smith",
        "subject" : "I like fish",
        "posted_date" : "2017-08-28",
        "tags" : ["fish", "ocean", "food"],
        "body" : "I like to cook and eat fish."
}
```


IBM Training

Cloudant API: Inserting a document - HTTP PUT or POST

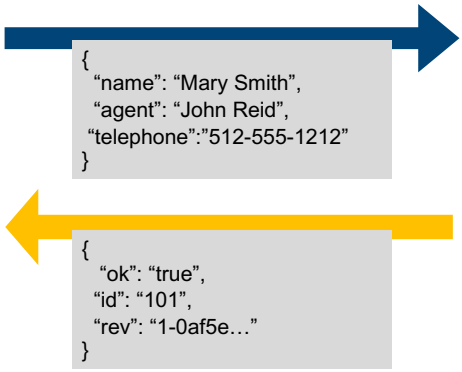
- using ***POST*** with *_id* in the document body:

```
POST https://<username>.cloudant.com/authors
```



- using ***PUT*** with *_id* in the URL:

```
PUT https://<username>.cloudant.com/authors/101
```

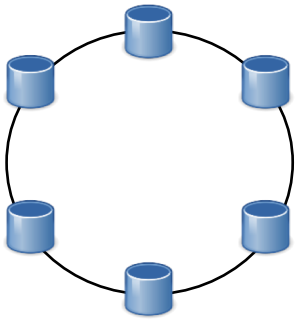


IBM Training

Cloudant API: Returning a single document

- HTTP GET with database name and _id of document
- For example, to get document with _id **100** from the **authors** database:

```
GET https://<username>.cloudant.com/authors/100
```

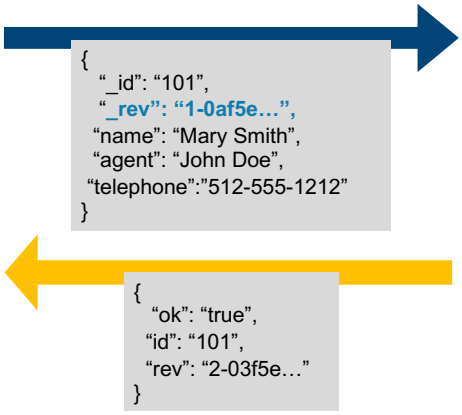


IBM Training

Cloudant API: Updating a document – HTTP PUT or POST

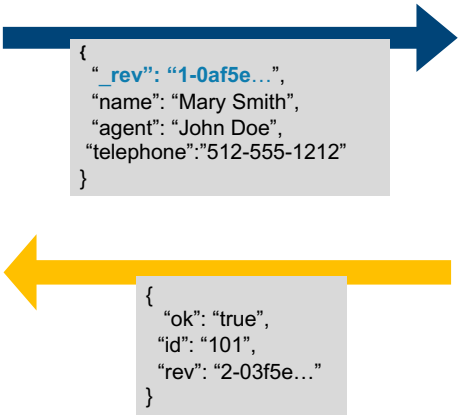
- Same HTTP operations as insert; If _id exists, it is an update
- Latest **_rev** is required or operation fails
- using ***POST*** with *_id* in the document body:

```
POST https://<username>.cloudant.com/authors
```



- using ***PUT*** with *_id* in the URL:

```
PUT https://<username>.cloudant.com/authors/101
```



IBM Training

Cloudant API: Deleting a document – HTTP DELETE or PUT

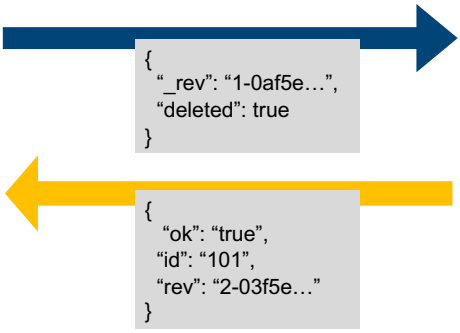
- using ***DELETE*** with *_id* and *_rev* in the document body:

```
DELETE
https://<username>.cloudant.com/authors/101/
?rev=...
```

- **_rev** is required or operation fails

using ***PUT*** with *_id* in the URL, *_rev* and `deleted:true` in document body :

```
PUT https://<username>.cloudant.com/authors/101
```



IBM Training

Cloudant documents: _id and _rev

- Each document has an **_id** (ID) field that is unique per database
  - Any string can be supplied as an _id, but it is recommended that you allow Cloudant to generate a UUID (universally unique identifier)

```
{
  "_id" : "7f123e23a328bd50ee123cd35452ae47",
  "_rev" : "2-3123414209",
  "title" : "IBM Cloudant Redbook",
  "author" : "Christopher Bienko"
}
```

- There is also a unique **_rev** (revision number) field per document
  - Generated by an md5 hash of the transport representation of the document
  - *N*-prefix reflects the number of times this document has been updated
  - Updates to existing documents must provide the latest _rev value
    - Otherwise the update request is rejected

IBM Training

Summary

- Think about how and where data is used
- Select the best cloud database service
- NoSQL databases do not all have the same capabilities
- Cloudant is a good choice for NoSQL

IBM Training

References (1 of 2)

Cloudant

- https://cloudant.com/learning-center/

“RDBMS to NoSQL: Reviewing Some Next-Generation Non-Relational Database's”

- http://liacs.leidenuniv.nl/~stefanovtp/courses/StudentenSeminarium/Papers/DB/3.IJAEST-Vol-No-11-Issue-No-1-RDBMS-to-NoSQL-Reviewing-Some-Next-Generation-Non-Relational-Database's-015-030.pdf

Eventual consistency

- http://guide.couchdb.org/draft/consistency.html

“NoSQL Distilled”

- http://www.amazon.com/dp/0321826620

Extract, Transform, and Load (ETL)

- https://en.wikipedia.org/wiki/Extract,_transform,_load

Graph database introduction

- https://www.compose.com/articles/graph-101-getting-started-with-graphs/

IBM Training

References (2 of 2)

JanusGraph concepts

- https://help.compose.com/docs/janusgraph-concepts

Introduction to graph concepts

- https://tinkerpop.apache.org/docs/3.2.3/reference/#intro

Compose for JanusGraph on IBM Cloud

- https://console.bluemix.net/docs/services/ComposeForJanusGraph/index.html#getting-started-with-compose-for-janusgraph

IBM Training

Developing Microservices

IBM Training

Prerequisites

- Microservices
  - Microservice architecture

IBM Training

Unit objectives

- Explain how to use microservices to design for failure
- Describe how microservices connect with one another
- List techniques for refactoring existing applications into microservices

IBM Training

Developing microservices

IBM Training

Developing a microservices application

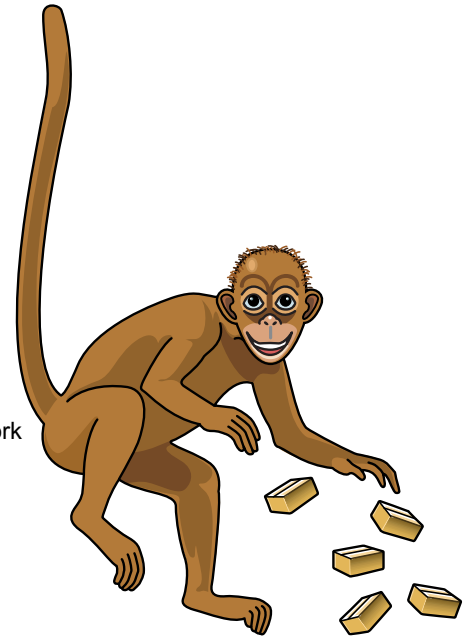
1. Identify a set of independent business tasks
   - Build initial microservices around those tasks
   - Teams that can work independently
   - Components that can be deployed, scale, and fail independently
2. Design for failure
   - Use microservices to make the application more robust
3. Design for scale
   - Service discovery
     - For example, Eureka
   - Configuration repositories
     - For example Zookeeper, Archaius
   - Common logging

- A service mesh is very helpful for steps 2 and 3

IBM Training

Design for failure

- Any service can fail
  - In a monolith, when one part stops working, it all stops working
  - Application must keep working and stay responsive
- Employ patterns for resiliency
  - **Service Registry**: Dynamic listing of service instances
  - **Circuit Breaker**: Block calls to a service that's not working
  - **Bulkhead**: Separate connection pools for separate resources
  - **Command**: Make requests easy to retry, throttle, and monitor
- Test for failure
  - Purposely, randomly cause problems
  - Example: Istio fault injection, Netflix OSS Simian Army framework



IBM Training

Service registry

- How service clients find service providers
  - Each service provider is a microservice instance
  - Client can use any instance of a microservice class
  - The pool of instances and their locations can change:
    - Elasticity
    - Health management
    - Load distribution
- Service registry
  - As service instances are started, they are listed in the registry
  - The registry knows the location of each service, even after it moves
  - The registry can distribute load across the instances
- Examples
  - Eureka, Consul
  - KubeDNS
  - CloudFoundry GO Router

IBM Training

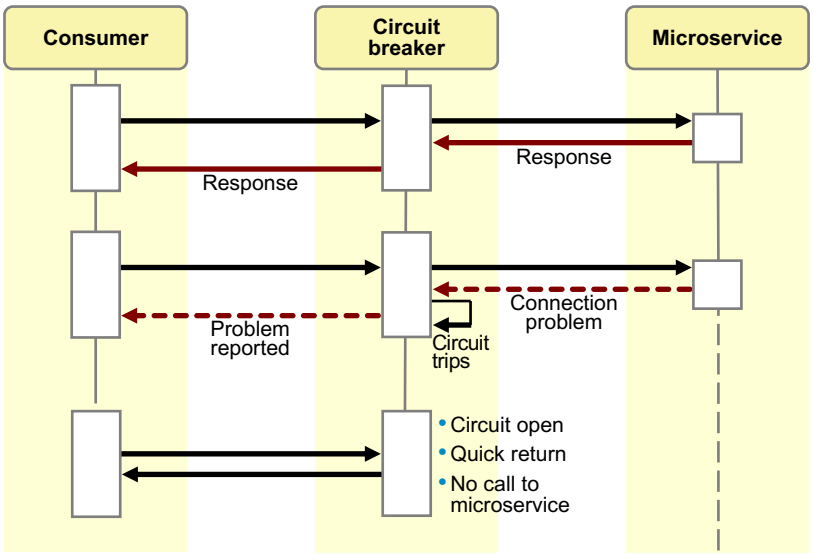
Service discovery in IBM Cloud compute models

- The Cloud Foundry router is the service registry
  - Each microservice class is a runtime with a route
  - Cloud Controller manages microservice instances
  - Router distributes load across the microservice instances
  - Blue-green deployment supports multiple versions of a microservice
- Kubernetes clients find the microservices in pods using Kube DNS and services
  - DNS enables finding services within namespaces
  - Each pod replica registers its IP address with its service
  - The service distributes load across the pod replicas
  - Istio Pilot provides platform agnostic service discovery interface
- For microservices deployed across regions, use an external registry
  - Examples:
    - Eureka and Zookeeper

IBM Training

Circuit breaker

- A service consumer is only as reliable as its service provider
- Avoid invoking an unreliable service instance
  - Service might throw an error
  - Service might time out
  - Network might fail
- Circuit breaker
  - Consumer can invoke reliably
  - Invokes unreliable service
  - Fails fast when the service isn't working



IBM Training

Command and circuit breaker

- Circuit breakers are typically used with command objects
- What happens if a circuit breaker fails?
  - How do you retry using a different service instance?
  - Every consumer shouldn't have to implement retry logic
- Command object
  - Wrap a service request as a command
  - A failed command can be retried
  - A command can implement a Plan B
  - Commands can be queued for throttling
  - Enable metrics to be measured for monitoring
- Example: Netflix Hystrix
  - Framework for making microservice invocations fault and latency tolerant
  - Classes and code for commands, circuit breakers, and bulkheads
  - Monitoring dashboard to view services' availability and performance
- Example: Istio Envoy failure handler

IBM Training

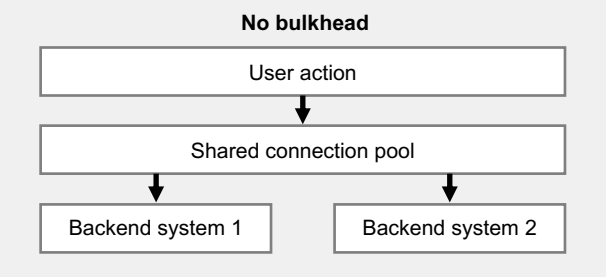
Bulkhead

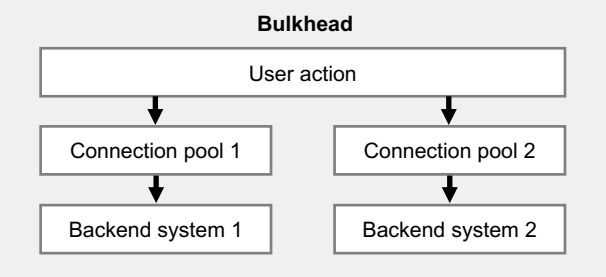
No bulkhead

- Single connection pool shared to connect to multiple external systems
- If one system blocks connections
  - All connections block on the one system
  - No way to connect to the working systems

Bulkhead

- Separate connection pool for each external system
- If one system blocks connections
  - All its connections block
  - Meanwhile, connections to the working systems still work
- Example: Hystrix or Istio





IBM Training

Microservices integration

IBM Training

Communication between services

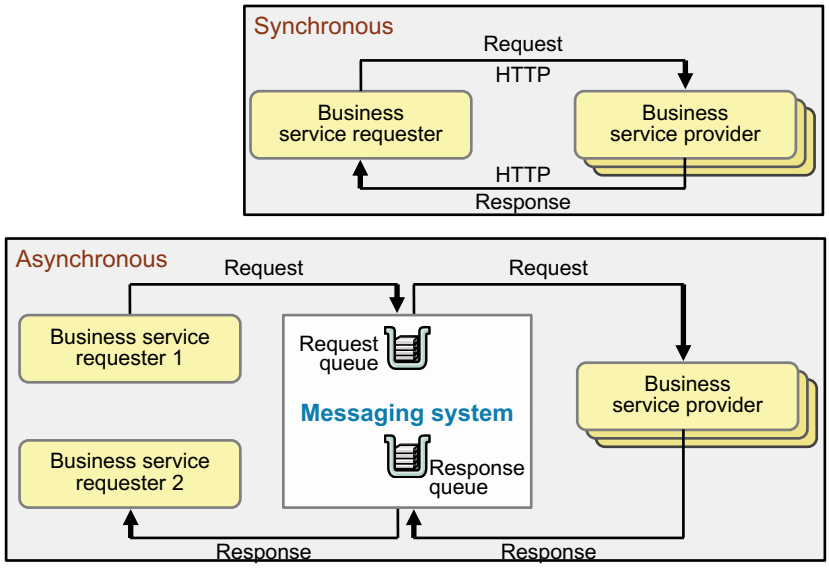
- Communication between services should be language-neutral
  - Services can be written in different languages
  - Cloud Foundry currently supports only inbound HTTP
- Most often REST synchronous protocol
  - JSON, HTTP
- For asynchronous protocol
  - Messaging system, typically AMQP or Kafka
  - JSON message payloads
  - Service activators to perform command messages synchronously
- Avoid alternative invocation styles
  - Leads to an explosion of functions and invocation styles

IBM Training

Asynchronous communication

Asynchronous communication can make microservices more robust

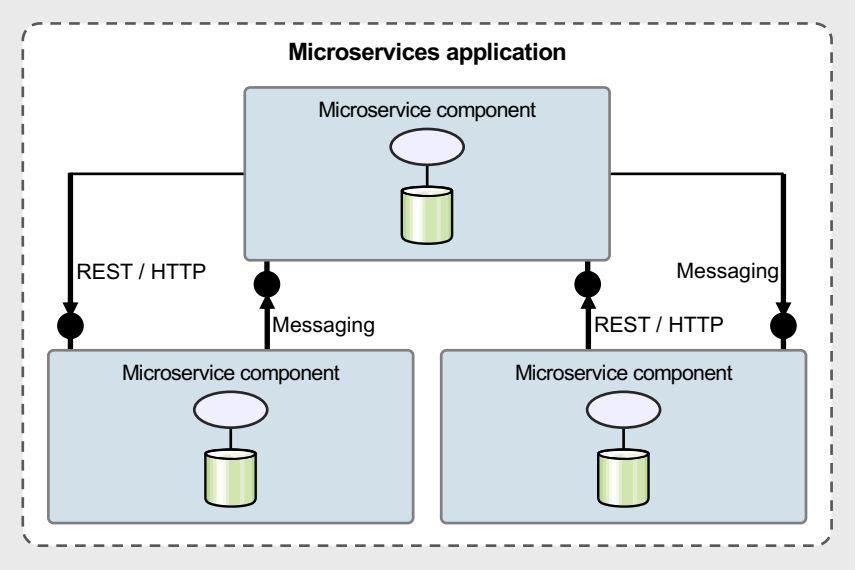
- Requester doesn't have to block while provider runs
- Different requester instance can handle response
- Messaging system holds action and result



IBM Training

Microservices intercommunication

- Lightweight protocols
  - REST such as JSON and HTTP
  - Messaging such as Kafka
- The aim is complete decoupling
  - Messaging wherever possible
  - Service registry or discovery
  - Load balancing
  - Circuit breaker patterns



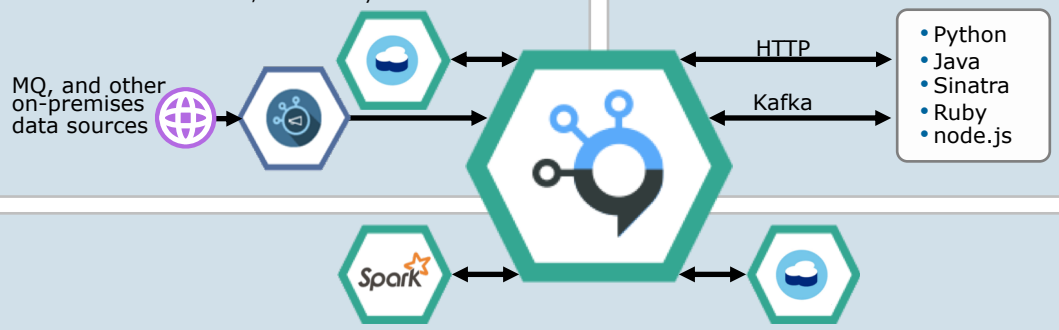
Developing Microservices 15

IBM Training

Message Hub service in IBM Cloud

Hub for asynchronously connecting services inside IBM Cloud or beyond
Applications that are connected to events that happen in other IBM Cloud services, or from beyond the cloud

Microservices enable applications to evolve
Open protocols support polyglot runtimes, application-controlled behavior, and reactive scale



Insights from the data you already have
Data needs to be streamed from anywhere to one or many analytics engines

Developing Microservices 16

IBM Training

Refactoring to microservices

IBM Training

Evolving to microservices

- Agile development: What's the simplest thing that could possibly work?
  - A monolith is simpler
- Start with a monolith
  - Make it modular, and plan that modules can become microservices
  - Each module should be a vertical slice, a mini-app with its own data
- Start with a minimally viable product (MVP)
  - MVP is a module – improvements to existing features go in the same module
  - Implement new functional tasks in new modules
  - When a module implements more than one task, refactor into separate modules
- Separate modules into microservices as needed
  - Multiple teams want to work independently
  - Monolithic code base is becoming unwieldy to maintain
  - Modules need to deploy, scale, or fail independently

IBM Training

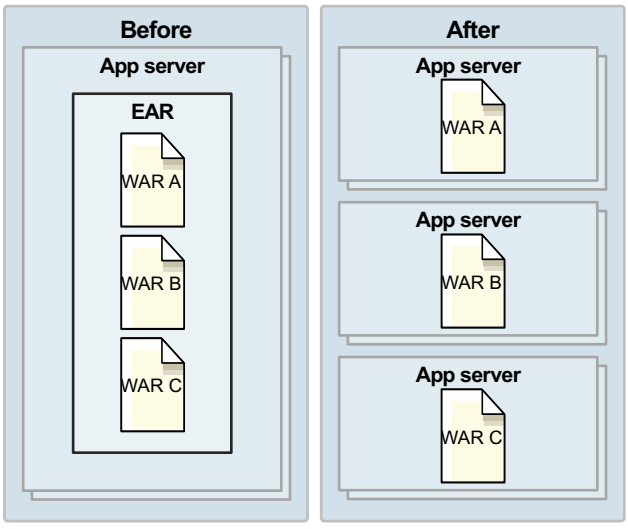
Refactoring an existing application

- Places to refactor
  - Each REST service is potentially a microservice
  - Each SOAP web service or EJB is potentially a microservice
    - Especially stateless session beans
    - Redesign function-oriented interfaces to asset-oriented interfaces
    - Make the interface RESTful, such as using command objects
  - Use domain-driven design to discover your assets, which might be microservices
- Refactor the data for the microservices code
  - Each microservice manages its own data
  - A set of tables that is only used by one module
  - A set of tables that is only used at a particular time, like a daily status summary
  - Refactor the tables so that each table is used by only one module
    - Optimize for query time, not storage efficiency

IBM Training

Repackaging an existing application

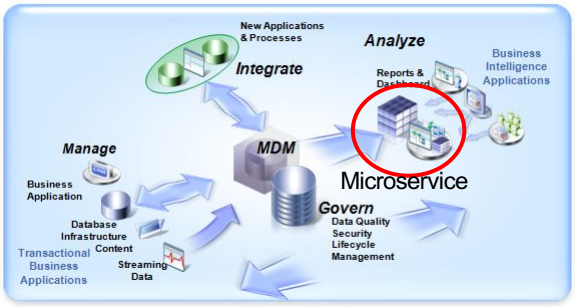
- Split up EAR files
  - Divide along functional lines
  - Package independent WAR files
  - Might require minor code changes
- Deploy WAR files separately
  - Its own Liberty for Java instant runtime
  - Its own Liberty Docker container
- Build, deploy, and manage separately
  - Use independent DevOps pipelines for each WAR file
  - Scale each separately and manage independently

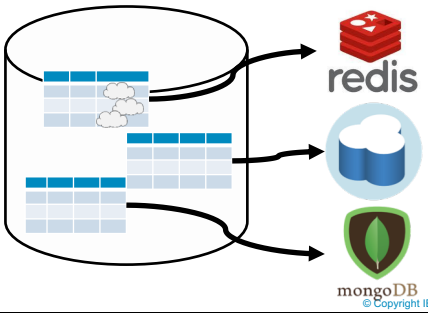


IBM Training

Refactoring data for microservices

- Master data management
  - Form a single, consistent view of widely used data entities
  - Develop microservices for working with that
- Code is storing blobs in your SQL database
  - Store those objects in a NoSQL database instead
  - Key-value store
    - Redis
- Active Record pattern: Flat objects unrelated to others
  - Islands of data unrelated to other data
  - Use a document model database
    - Cloudant or Mongo





IBM Training

Configuration for microservices

Make configuration part of your DevOps process

- Treat infrastructure as code, software-defined data center
- All application deployment must be automated
- Start small with simple scripts and build from there

IBM Training

Conclusion

IBM Training

Unit summary

- Explain how to use microservices to design for failure
- Describe how microservices connect with one another
- List techniques for refactoring existing applications into microservices

IBM Training

Next

- More on microservices
  - Service mesh
- Deploying cloud applications
  - Docker and Kubernetes in IBM Cloud Container Service
  - Cloud Foundry in IBM Cloud
  - Swarm in Docker Enterprise Edition for IBM Cloud
  - OpenWhisk in IBM Cloud Functions
- IBM Cloud capabilities
  - IBM Cloud architecture
  - WebSphere Liberty for microservices using Java, Java EE, and MicroProfile
  - Cloud data services
  - Integration with systems of record
- DevOps
  - Continuous delivery of microservices

IBM Training

Resources

- *Microservices from Theory to Practice: Creating Applications in IBM Bluemix Using the Microservices Approach*, IBM Redbook, 26 August 2015, http://bit.ly/ibmredmicro
- "Microservices, SOA, and APIs: Friends or enemies?" by Kim J. Clark, *IBM developerWorks*, 21 January 2016
- "Microservices.TV" https://developer.ibm.com/tv/microservices/
- "Microservices in action, Part 1: Introduction to microservices" by Rick Osowski, *IBM developerWorks*, 26 June 2015
- "Introduction to Microservices and Cloud Native Application Architecture" by David Currie
- "Microservices" by James Lewis and Martin Fowler, http://bit.ly/microsvc
- *Building Microservices: Designing Fine-Grained Systems* by Sam Newman, O'Reilly Media, February 2015, http://bit.ly/oreillymicro
- "Netflix Hystrix: How It Works" https://github.com/Netflix/Hystrix/
- Spring Cloud https://cloud.spring.io
- *Release It!: Design and Deploy Production-Ready Software* by Michael T. Nygard, Pragmatic Bookshelf, April 2007
- *Domain-Driven Design: Tackling Complexity in the Heart of Software* by Eric Evans, Addison-Wesley Professional, August 2003
- *Enterprise Integration Patterns* by Gregor Hohpe and Bobby Woolf, Addison-Wesley Professional, October 2003

IBM Training

Service Mesh

IBM Training

Prerequisites

- Microservices
  - Microservice architecture
  - Microservice development

IBM Training

Agenda

- Why service mesh?
- Common functions of a service mesh
- Services mesh implementations

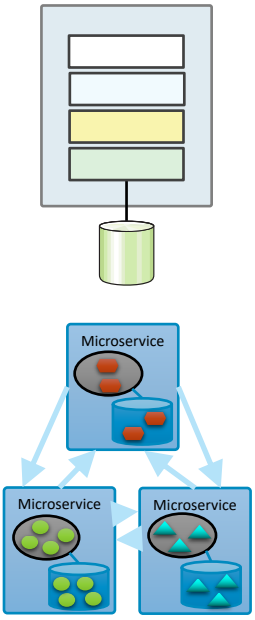
IBM Training

Why Service Mesh?

IBM Training

Why Service Mesh?

- In monolithic applications, the location of resources used are:
  - Well-known
  - Relatively static
  - Found in configuration files or hard-coded



- This is not true for a microservices implementation
- Required services could be anywhere
  - Cloud providers, public or private, datacenter, VMs, containers, servers
  - Service instances have dynamic locations
    - Come and go because of elasticity, failures, upgrades

Service Mesh 5

IBM Training

The complexity tradeoff

Faster Independent Deployments

Increased Operational Complexity

Service Mesh 6

IBM Training

What are the challenges with microservices?

- How can I find the service I need?
  - What if it moves or fails?
- How can I scale my services?
  - Spread my load against multiple instances?
  - Can I control how I spread the load?
- How can I test new versions of services?
  - Without impacting existing users
- How can I test against failures?
  - How does the failure of one service affect the others?
  - Do my services have reasonable timeouts?
  - Can I avoid cascading failures?

Service Mesh 7

IBM Training

A service mesh can help

- These are common challenges for everyone creating microservices
- Don't reinvent the wheel
- Variety of solutions using open source tools

NETFLIX OSS

Istio

CONSUL by HashiCorp

lagom

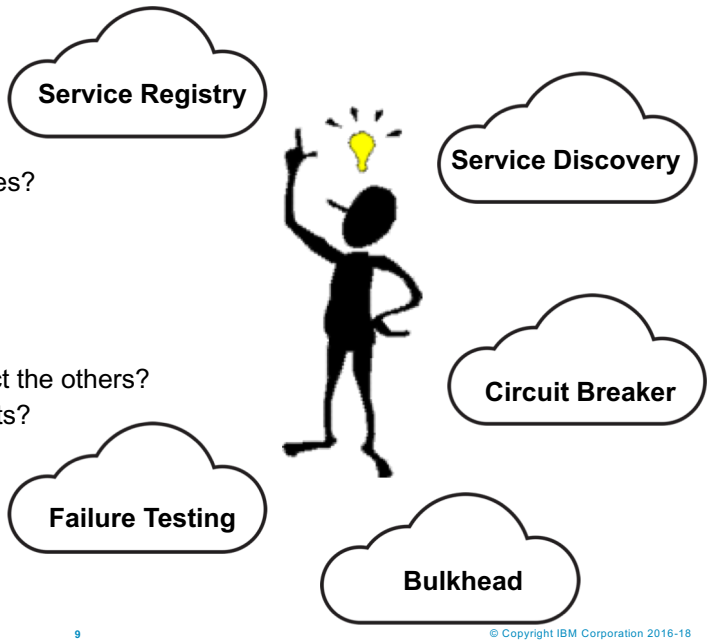
airbnb

Service Mesh 8

IBM Training

A service mesh can help

- How can I find the service I need?
  - What if it moves or fails?
- How can I scale my services?
  - Spread my load against multiple instances?
  - Can I control how I spread the load?
- How can I test new versions of services?
  - Without impacting existing users
- How can I test against failures?
  - How does the failure of one service affect the others?
  - Do my services have reasonable timeouts?
  - Can I avoid cascading failures?



Service Mesh 9

IBM Training

Common Functions of a Service Mesh

10

IBM Training

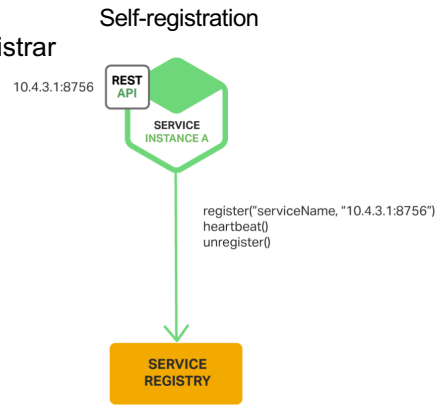
What is a service mesh?

- A network for services, not bytes
  - A domain specific router for microservice interactions
  - Makes service-to-service communication work better
- Distributed security
  - Service-to-service authentication and encryption – TLS
- Intelligent traffic management
  - Resiliency – Failure handling: circuit breaker, deadlines, retry
  - Routing – Routing rules, service discovery, load balancing, latency
  - Policy enforcement – Shared configuration, uniform communication
- Visibility
  - Distributed tracing – Track each request through the microservices
  - Instrumentation – Metrics and monitoring
  - Dashboards – See the microservices instances and the traffic through them

IBM Training

Service Registry

- Database containing the network locations of service instances
- Should be highly available
- Instances self-register or are registered through a 3rd party registrar
- Heartbeat confirms instances are still alive



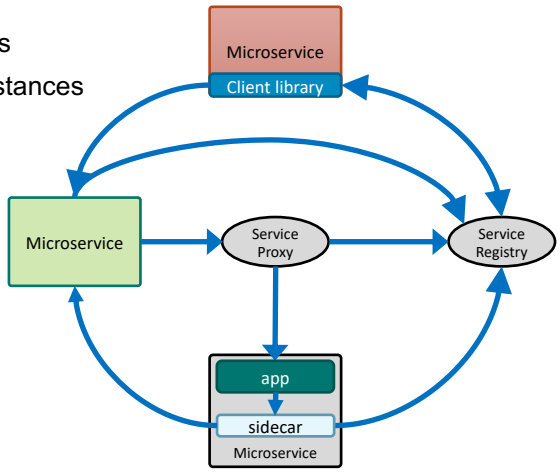
Examples:
- Netflix Eureka
- etcd (Used by Kubernetes & Cloud Foundry)
- Consul
- Apache Zookeeper

Source: dzone.com

IBM Training

Service Discovery / Service Proxy

- How client services find & connect to provider services
- Each provider service provider could have multiple instances
- The pool of instances and their locations can change:
  - Elasticity
  - Health management
  - Load distribution
- 2 Models
  - Client-side
  - Server-side

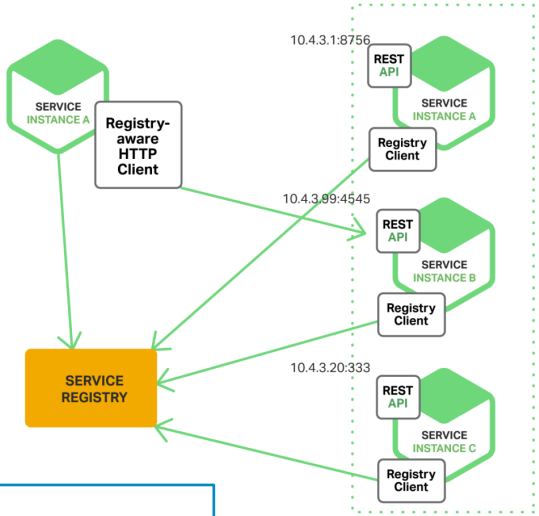


Service Mesh 13 © Copyright IBM Corporation 2016-18

IBM Training

Client-Side Discovery

- Client code is responsible for:
  - Finding locations of available service instances
  - Load balancing logic for requests
- Client queries a service registry for location of service instances
- Client uses its own load balancing algorithm to select which instance to request
- Pros
  - Less components
  - Client can make application-specific load distribution decisions
- Cons
  - Client load distribution logic needs to be developed for every language used
  - Logic changes need to be in code



Examples:
- Netflix Eureka Service Registry
- Netflix Ribbon IPC Client

Source: dzone.com

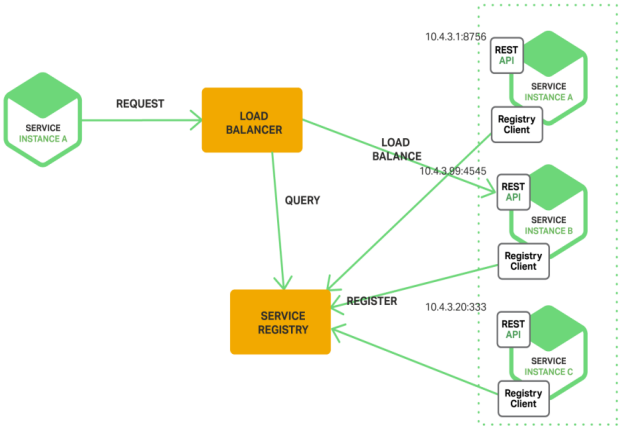
Service Mesh 14 © Copyright IBM Corporation 2016-18

IBM Training

Server-Side Discovery

- Client makes request to a load balancer
  - Load balancer queries service registry and routes requests to instances
  - Distribution logic in load balancer
- Pros
  - Client does not need discovery & load balancing logic
- Cons
  - Unless the load balancer is provided by the mesh, it is another component to manage

Examples:
- AWS Elastic Load Balancer (ELB)
- NGINX / NGINX Plus Http Server & Load Balancer
- Istio



Source: dzone.com

Service Mesh 15

IBM Training

Service discovery in IBM Cloud compute models

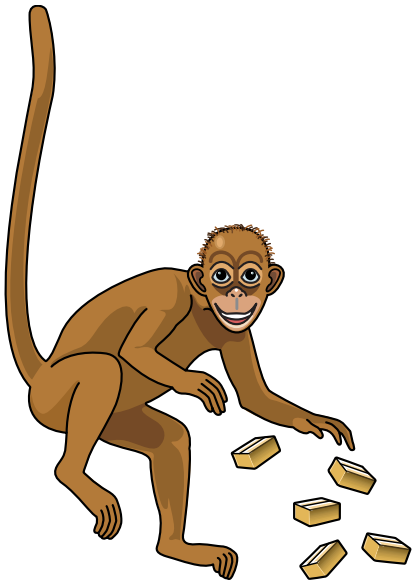
- The Cloud Controller and Router in Cloud Foundry provide service registry and service discovery
  - Each microservice class is a runtime with a route
  - Cloud Controller manages microservice instances
  - Router distributes load across the microservice instances
  - Blue-green deployment supports multiple versions of a microservice
- Kubernetes clients find the microservices in pods using Kube DNS and services
  - DNS enables finding services within namespaces
  - Each pod replica registers its IP address with its service
  - The service distributes load across the pod replicas

Service Mesh 16

IBM Training

Automated Testing

- Test for failure
  - Purposely, randomly cause problems
- Employ patterns for resiliency
  - Service Registry: Dynamic listing of service instances
  - Circuit Breaker: Block calls to a service that's not working
  - Bulkhead: Separate connection pools for separate resources
  - Command: Make requests easy to retry, throttle, and monitor



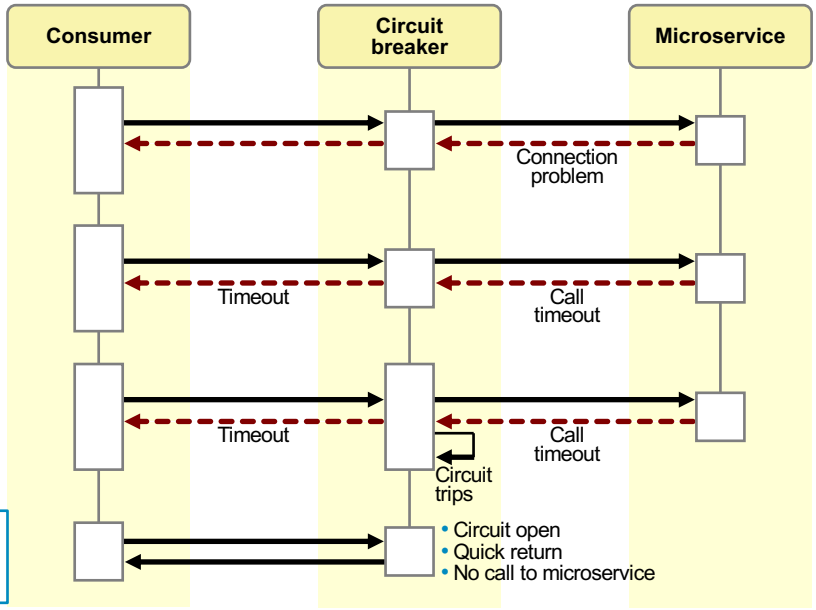
Examples:
- Netflix Chaos Monkey, Simian Army
- Istio

Service Mesh 17

IBM Training

Circuit breaker

- A service consumer is only as reliable as its service provider
- Avoid invoking an unreliable service instance:
  - Service might throw an error
  - Service might time out
  - Network might fail
- Circuit breaker
  - Consumer can invoke reliably
  - Invokes unreliable service
  - Fails fast when the service isn't working



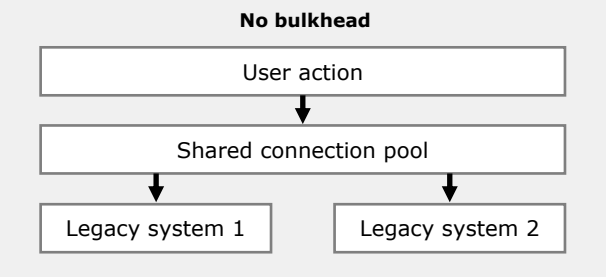
Example:
- Netflix Hystrix
- Istio

Service Mesh 18

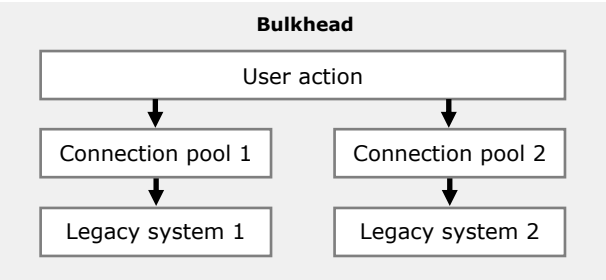
IBM Training

Bulkhead

- No bulkhead
  - Single connection pool shared to connect to multiple external systems
  - If one system blocks connections
    - All connections block on the one system
    - No way to connect to the working systems



- Bulkhead
  - Separate connection pool for each external system
  - If one system blocks connections
    - All its connections block
    - Meanwhile, connections to the working systems still work



Example:
- Netflix Hystrix
- Istio

Service Mesh 19

IBM Training

Services Mesh Implementations

Netflix OSS

Istio

20

What is Netflix OSS?

- **Netflix OSS** – *A software-based control plane that is portable across clouds, can be run the same locally on a laptop or in any number of public Cloud providers.*

- Netflix is the “poster child” of microservices
  - Netflix OSS is open source of many of the tools and methods they have used to build and improve their site
  - Over 33 root-level projects (and more actively maintained)
  - Most are Java libraries – Class structure could restrict you to only running your code in their framework
- Netflix OSS components available via https://netflix.github.io

IBM Training

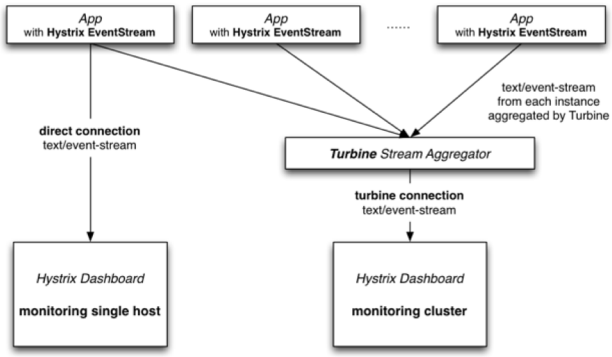
Major Netflix OSS Projects

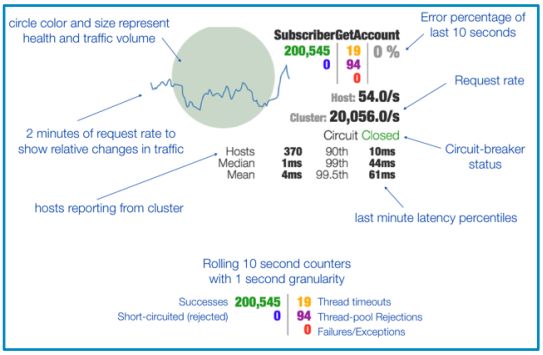
- Eureka, Ribbon, Zuul – Service Registry / Discovery / Proxy
  - Client-side discovery – logic built into application libraries
  - Many applications automatically integrate with Eureka (or are enabled to through the use of Spring Cloud enhancements)
- Chaos Monkey, Simian Army – Failure Testing
  - Chaos Monkey was the first project
  - Simian Army includes Chaos Monkey and an "army" of other monkeys
    - Latency Monkey, Conformity Monkey, Doctor Monkey, Janitor Monkey, Security Monkey, and more
  - Programmatically control what parts of their infrastructure are unstable, unavailable, and unreliable
    - See how this impacts overall system health.
  - Original Chaos Monkey was built to the AWS APIs
    - Other parts of the Simian Army not written to specific provider APIs

IBM Training

Netflix OSS Circuit Breaker – Hystrix, Turbine, Hystrix Dashboard

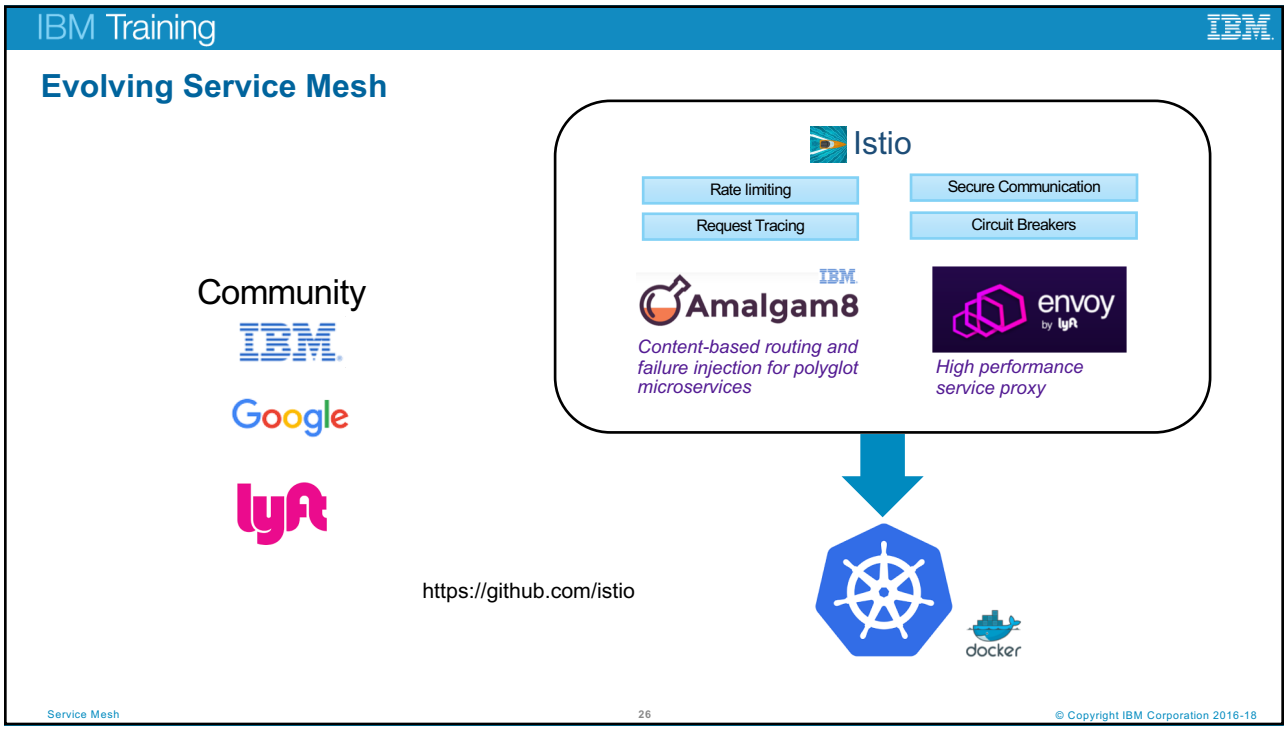
- Java-based implementation of the Circuit Breaker pattern
- Provides reliability beyond individual service calls & visibility into hot-spots / bottlenecks
- Isolates latency and increases fault tolerance at runtime
- Turbine aggregates clustered services into a single stream for visibility through Hystrix Dashboards





Source: techblog.netflix.com

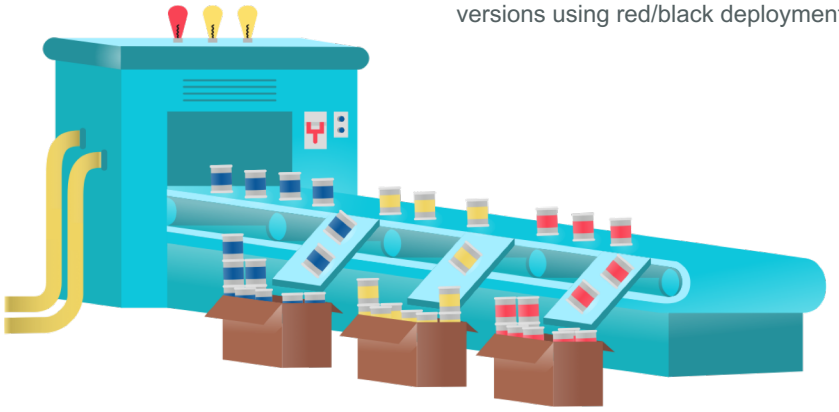

IBM Training
IBM
Istio
http://istio.io
An open platform to connect, manage, and secure microservices
Service Mesh
25
© Copyright IBM Corporation 2016-18


IBM Training
IBM
Evolving Service Mesh
Community
IBM
Google
lyft
https://github.com/istio
Istio
Rate limiting
Secure Communication
Request Tracing
Circuit Breakers
IBM
Amalgam8
Content-based routing and failure injection for polyglot microservices
envoy by lyft
High performance service proxy
docker
Service Mesh
26
© Copyright IBM Corporation 2016-18

IBM Training

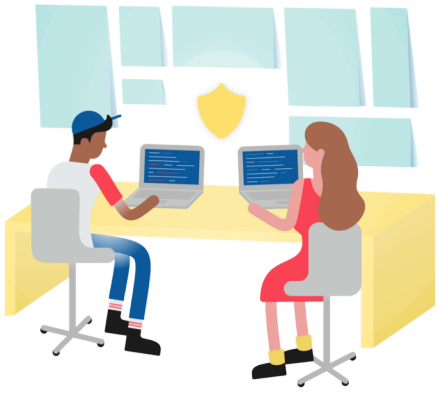
Intelligent Routing and Load Balancing

Control traffic between services with dynamic route configuration, conduct A/B tests, release canaries, and gradually upgrade versions using red/black deployments



IBM Training

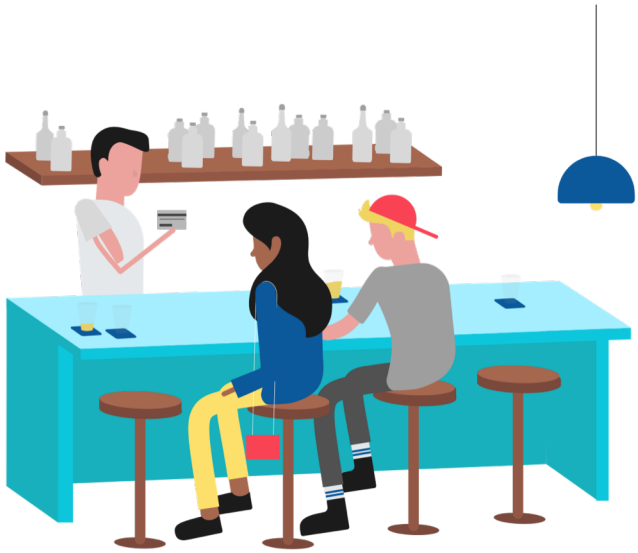
Resilience Across Languages and Platforms



Increase reliability by shielding applications from flaky networks and cascading failures in adverse conditions

IBM Training

Fleet Wide Policy Enforcement



Apply organizational policy to the interaction between services, ensure access policies are enforced and resources are fairly distributed among consumers

IBM Training

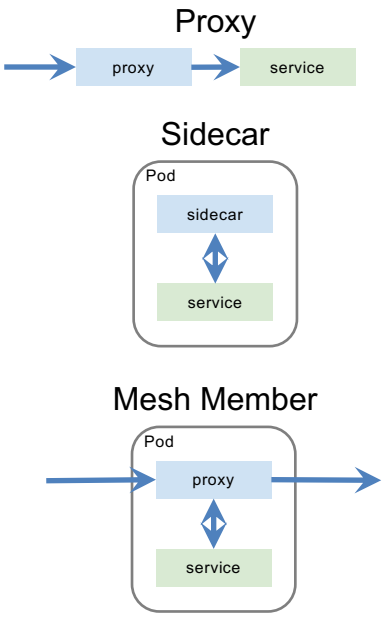
In-Depth Telemetry and Reporting

Understand the dependencies between services, the nature and flow of traffic between them and quickly identify issues with distributed tracing

IBM Training

Managing services: Proxy and sidecar

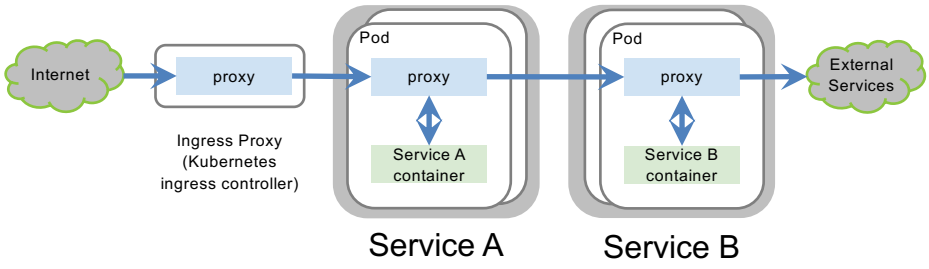
- A service mesh manages the interactions between services
  - It uses proxies, usually deployed as sidecars in pods
- Proxy – Controls access to another object
  - A service mesh uses proxies between services (and also clients)
  - Enables the mesh to manage interactions
- Pod – Kubernetes deploys a set of containers as a pod
  - All of the containers in a pod are hosted on the same node
  - A pod can manage the network access to the containers in the pod
  - A pod typically hosts a single container, but can host multiple as a unit
- Sidecar – Adds behavior to a container without changing it
  - Sidecar and service behave as a single enhanced unit
  - Pod hosts them as a single unit
- Mesh Member – Access to the service goes through the proxy
  - Deploys the proxy as a sidecar in the same pod
  - Configures the pod so that all network access goes through the proxy



Service Mesh 31 © Copyright IBM Corporation 2016-18

IBM Training

Mesh Request Flow



Inbound features
- Service authentication
- Authorization
- Rate limits
- Load shedding
- Telemetry
- Request Tracing
- Fault Injection

Outbound features
- Service authentication
- Load balancing
- Circuit breaker and retry
- Fine-grained routing
- Telemetry
- Request Tracing
- Fault Injection

Service Mesh 32 © Copyright IBM Corporation 2016-18

IBM Training

Envoy: Istio's proxy of choice

- A C++ based L4/L7 proxy
- Low memory footprint
- Battle-tested @ Lyft
  - 100+ services
  - 10,000+ VMs
  - 2M req/s
- Now a Cloud Native Computing Foundation (CNCF) open source project
  - Not unilaterally controlled by Lyft or Istio

Goodies:

- HTTP/2 & gRPC
- Zone-aware load balancing w/failover
- Health checks, circuit breakers, timeouts, retry budgets
- No hot reloads - API driven config updates

Istio's contributions:

- Transparent proxying w/SO_ORIGINAL_DST
- Traffic routing and splitting
- Request tracing using Zipkin
- Fault injection

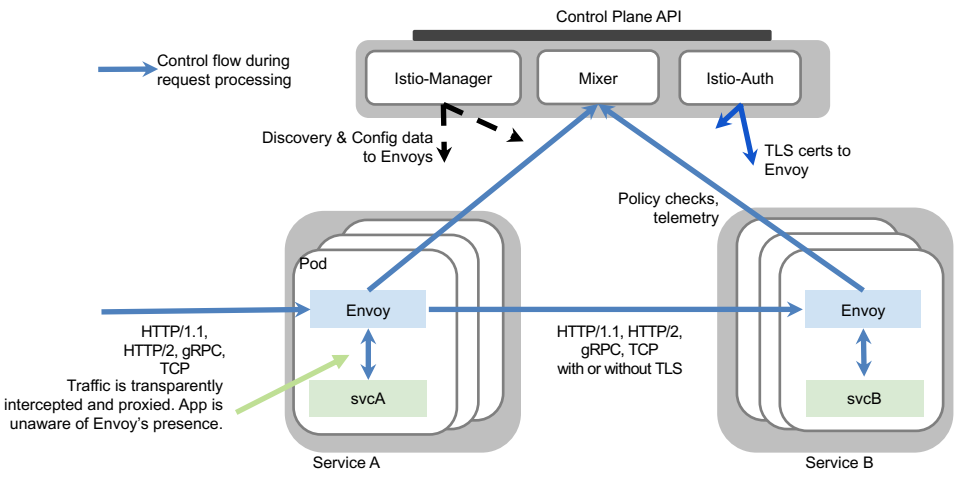
Service Mesh 33 © Copyright IBM Corporation 2016-18

IBM Training

Istio Architecture

Control Plane

Data Plane



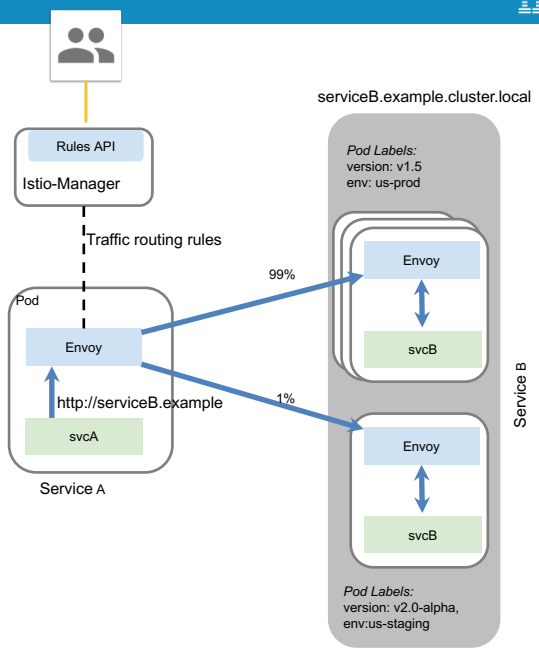
Service Mesh 34 © Copyright IBM Corporation 2016-18

IBM Training

Traffic Splitting

```
// A simple traffic splitting rule
destination: serviceB.example.cluster.local
match:
  source: serviceA.example.cluster.local
route:
- tags:
    version: v1.5
    env: us-prod
  weight: 99
- tags:
    version: v2.0-alpha
    env: us-staging
  weight: 1
```

Traffic control is decoupled from infrastructure scaling

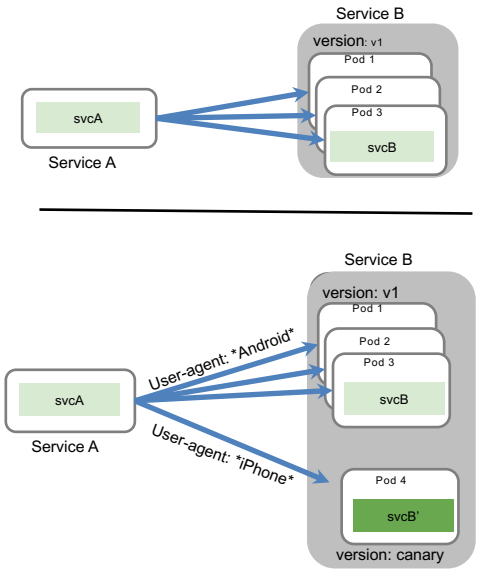


IBM Training

Traffic Steering

```
// Content-based traffic steering rule
destination: serviceB.example.cluster.local
match:
  httpHeaders:
    user-agent:
      regex: ^(.*?;)?(iPhone)(;.*)?$
precedence: 2
route:
- tags:
    version: canary
```

Content-based traffic steering



IBM Training

Resiliency

Istio adds fault tolerance to your application without any changes to code

```
// Circuit breakers
destination: serviceB.example.cluster.local
policy:
- tags:
    version: v1
  circuitBreaker:
    simpleCb:
      maxConnections: 100
      httpMaxRequests: 1000
      httpMaxRequestsPerConnection: 10
      httpConsecutiveErrors: 7
      sleepWindow: 15m
      httpDetectionInterval: 5m
```

Resilience features

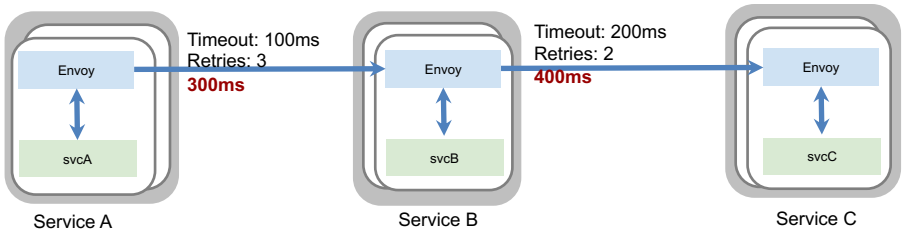
- Timeouts
- Retries with timeout budget
- Circuit breakers
- Health checks
- AZ-aware load balancing w/ automatic failover
- Control connection pool size and request load
- Systematic fault injection

Service Mesh 37

IBM Training

Resiliency Testing

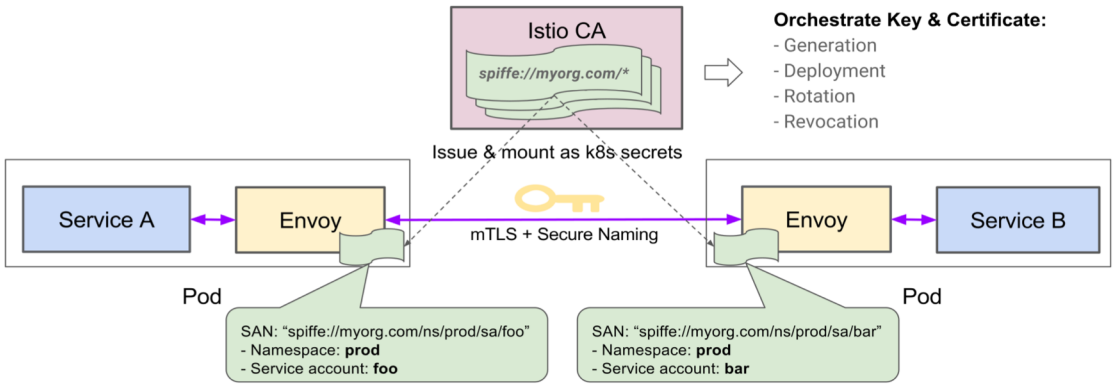
- Systematic fault injection to identify weaknesses in failure recovery policies
  - HTTP/gRPC error codes
  - Delay injection



Service Mesh 38

IBM Training

Istio Auth Service Authentication Architecture



spiffe.io

IBM Training

Conclusion

IBM Training

Conclusion

- Why service mesh?
- Common functions of a service mesh
- Services mesh implementations

IBM Training

Next

- Deploying cloud applications
  - Docker and Kubernetes in IBM Cloud Container Service
  - Cloud Foundry in IBM Cloud
  - Swarm in Docker Enterprise Edition for IBM Cloud
  - OpenWhisk in IBM Cloud Functions
- IBM Cloud capabilities
  - IBM Cloud architecture
  - WebSphere Liberty for microservices using Java, Java EE, and MicroProfile
  - Cloud data services
  - Integration with systems of record
- DevOps
  - Continuous delivery of microservices

IBM Training

Resources

- IBM, Google and Lyft give microservices a ride on the Istio Service Mesh
  - https://developer.ibm.com/dwblog/2017/istio/
- Istio
  - https://istio.io
  - https://github.com/istio
- developerWorks Container Orchestration Journeys
  - http://ibm.biz/kube-journeys
  - https://developer.ibm.com/code/journey/manage-microservices-traffic-using-istio
- Microservices
  - http://kasunpanorama.blogspot.com/2015/11/microservices-in-practice.html
- Service discovery
  - https://dzone.com/articles/service-discovery-in-a-microservices-architecture
- Service mesh
  - https://blog.buoyant.io/2017/04/25/whats-a-service-mesh-and-why-do-i-need-one/
  - https://content.pivotal.io/blog/pivotal-and-istio-advancing-the-ecosystem-for-microservices-in-the-enterprise
- Sidecars
  - https://blog.buoyant.io/2016/06/17/small-memory-jvm-techniques-for-microservice-sidecars/

IBM Training

Microservices traffic management and telemetry using Istio

Optimizing and visualizing the network in your microservice application

IBM Training

Topics

- Why Istio?
- Architecture
- Traffic management
  - Rule configuration
  - Traffic routing
  - Bookinfo example
- Policies and telemetry
- Debugging Istio

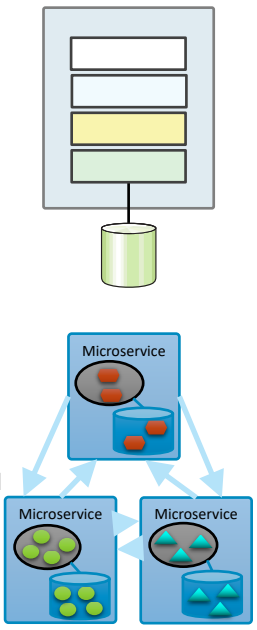
IBM Training

Why Istio

IBM Training

What is a service mesh?

- Service mesh
  - The network of microservices that make up an application and the interactions between them
  - A network of services, not bytes
- Why do we need a service mesh?
  - As a mesh gets bigger, it's harder to understand and manage
  - Manage the mesh as a whole, not as its parts
  - Provide behavioral insights and operational control
- New requirements
  - *Quality of service*: discovery, load balancing, failure recovery, rate limiting, metrics, and monitoring
  - *Continuous deployment*: canary rollouts, A/B testing, staged rollouts with %-based traffic splits, etc.
  - *Security*: end-to-end authentication and access control



IBM Training

What is Istio?

- An open platform to connect, secure, control and observe microservices
- A language-neutral service mesh for modern cloud-native applications
- Istio provides an easy way to create a network of deployed services
  - Load balancing, service-to-service authentication, monitoring, and more
  - Does not require any changes to the service code
- Istio uses a special service proxy throughout your environment
  - Intercepts all network communication between microservices
  - Configured and managed using Istio's control plane functionality

IBM Training

Why use Istio?

Intelligent Routing and Load Balancing
Control traffic between services with dynamic route configuration, conduct A/B tests, release canaries, and gradually upgrade versions using red/black deployments

Resilience Across Languages and Platforms
Increase reliability by shielding applications from flaky networks and cascading failures in adverse conditions

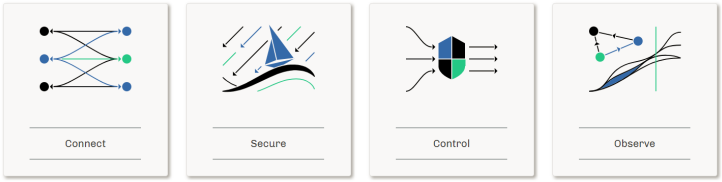
Fleet Wide Policy Enforcement
Apply organizational policy to the interaction between services, ensure access policies are enforced and resources are fairly distributed among consumers

In-Depth Telemetry and Reporting
Understand the dependencies between services, the nature and flow of traffic between them and quickly identify issues with distributed tracing

IBM Training

Core capabilities

- **Traffic Management**
  - Intelligently control the flow of traffic and API calls between services
  - Make calls more reliable and make the network more robust in the face of adverse conditions
- **Service Identity and Security**
  - Provide services in the mesh with a verifiable identity
  - Provide the ability to protect service traffic as it flows over networks of varying degrees of trustability
- **Policy Enforcement**
  - Apply organizational policy to the interaction between services
  - Ensure access policies are enforced and that resources are fairly distributed among consumers
  - Make policy changes by changing the configuration of the mesh, not the application code
- **Telemetry**
  - Gain understanding of the dependencies between services and the nature and flow of traffic between them
  - Provide the ability to quickly identify issues



IBM Training

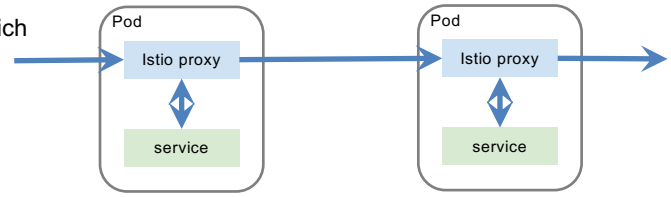
Istio Architecture

IBM Training

Istio advantages

- **Language-neutral**
  - Istio implementation is performed outside of the application container
  - Does not matter what programming language the application is written in

- **Transparent/Non-invasive**
  - Istio intercepts network traffic outside of the application
  - Masking the application network connection which is not detected by the application

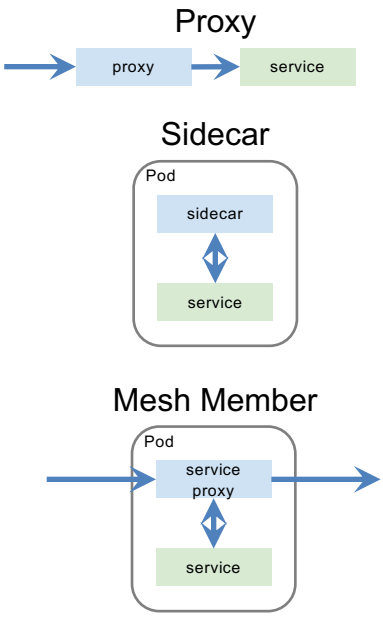


Microservices traffic management and telemetry using Istio 9

IBM Training

Managing services: Proxy and sidecar

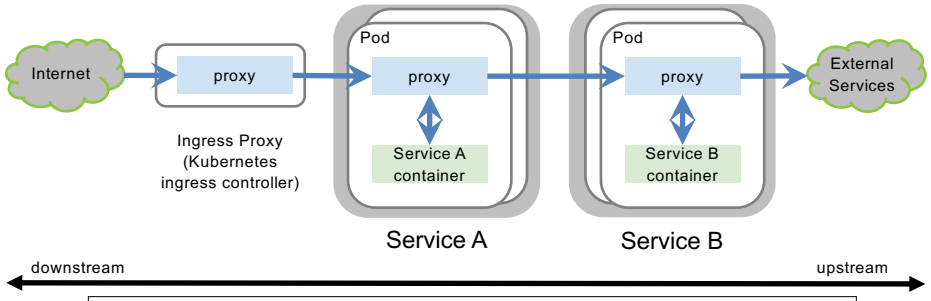
- A service mesh manages the interactions between services
  - It uses *proxies*, usually deployed as *sidecars* in *pods*
- **Proxy** – Controls access to another object
  - A service mesh uses proxies between services (and also clients)
  - Enables the mesh to manage interactions
- **Pod** – Kubernetes deploys a set of containers as a pod
  - All of the containers in a pod are hosted on the same node
  - A pod can manage the network access to the containers in the pod
  - A pod typically hosts a single container, but can host multiple as a unit
- **Sidecar** – Adds behavior to a container without changing it
  - Sidecar and service behave as a single enhanced unit
  - Pod hosts them as a single unit
- **Mesh Member** – Controlled access to the service
  - Deploys the *proxy* as a *sidecar* in the same *pod*
  - Configures the pod so that all network access goes through the proxy



Microservices traffic management and telemetry using Istio 10

IBM Training

Mesh Request Flow



What Each Proxy Adds

Inbound features	Outbound features
• Service authentication	• Service authentication
• Authorization	• Load balancing
• Rate limits	• Circuit breaker and retry
• Load shedding	• Fine-grained routing
• Telemetry	• Telemetry
• Request tracing	• Request tracing
• Fault injection (testing)	• Fault injection (testing)

IBM Training

Envoy: Istio's proxy of choice

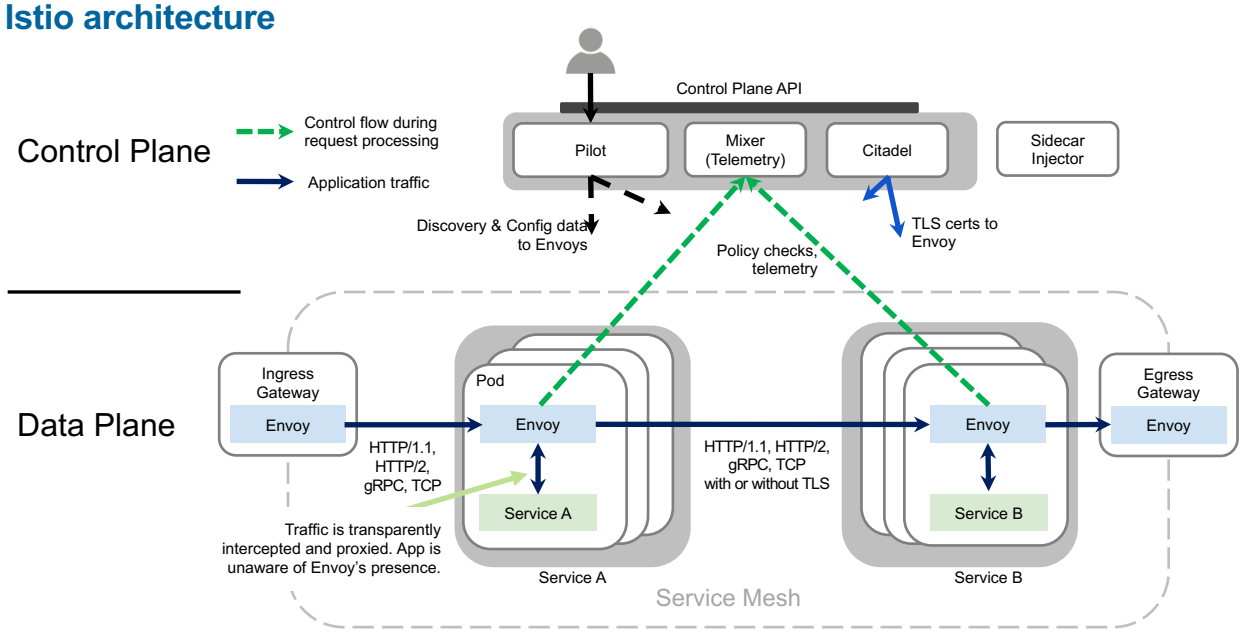
- A C++ based L4/L7 proxy
- Low memory footprint
- Battle-tested @ Lyft
  - 100+ services
  - 10,000+ VMs
  - 2M req/s
- Now an open source community project
  - Cloud Native Computing Foundation (CNCF)
  - Not unilaterally controlled by Lyft or Istio

Goodies

- Dynamic service discovery
- Load balancing
- TLS termination
- HTTP/2 and gRPC proxies
- Circuit breakers
- Health checks
- Staged rollouts with %-based traffic split
- Fault injection
- Rich metrics

IBM Training

Istio architecture



IBM Training

Istio architecture components

- **Envoy**
  - Service proxy that runs alongside an application container as a sidecar, intercepting incoming and outgoing communication
  - Gathers signals about traffic behavior as attributes, feeds them to the Mixer
- **Pilot**
  - Provides service discovery for the Envoy sidecars (built on the platform's service discovery for the application containers)
  - Configures traffic management capabilities for intelligent routing (e.g., A/B tests, canary deployments, etc.)
  - Configures resiliency (timeouts, retries, circuit breakers, etc.)
  - Converts routing rules that control traffic behavior into Envoy-specific configurations
- **Mixer**
  - Collects telemetry data from the Envoy proxy and other services
  - Enforces access control and usage policies across the service mesh
  - Plugin model interfaces with a variety of host environments and infrastructure backends
  - Istio 1.0 splits Mixer into policy and telemetry servers
- **Citadel**
  - Provides service-to-service and end-user authentication with built-in identity and credential management
  - Operators can enforce policies based on service identity rather than on network controls

IBM Training

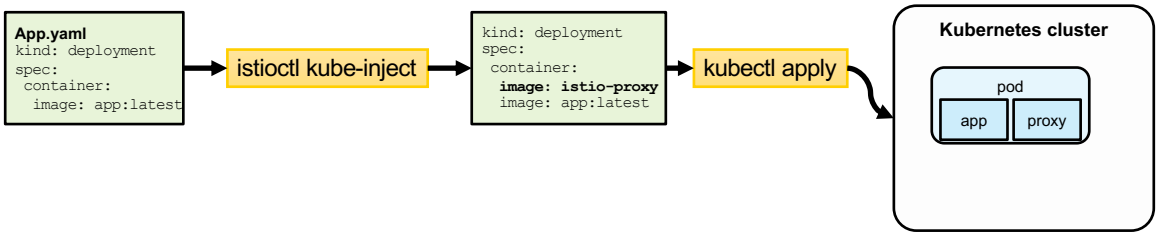
Additional Istio architecture components

- **Ingress gateway**
  - Handles incoming traffic to the mesh from external clients
- **Egress gateway**
  - Handles outgoing traffic from the mesh to external services
- **Sidecar injector**
  - Functionality to add Envoy proxies to deployed application containers

IBM Training

Manual Istio sidecar injection

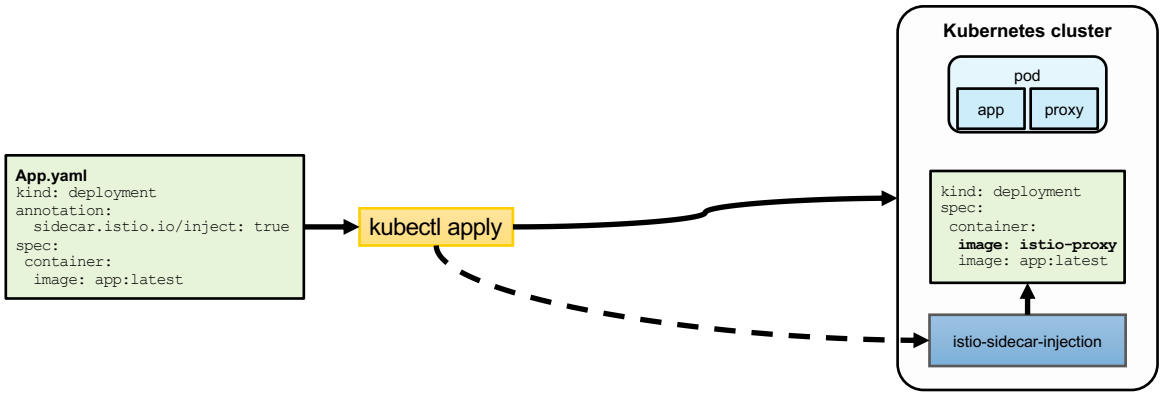
- To configure the service mesh, Istio must add its Envoy proxy to each of the services it will manage
  - This addition is performed by configuring Kubernetes using YAML file injection
- Manual injection
  - Run the command `istioctl kube-inject` on the YAML file
  - Resources are then created based on the modified manifest
  - `istioctl kube-inject -f app.yaml | kubectl apply -f -`



IBM Training

Automatic Istio sidecar injection

- Automatic injection
  - Either add a label to the namespace: `istio-injection=enabled`
  - Or annotate the pod or deployment with: `sidecar.istio.io/inject: "true"`
  - Kubernetes invokes `istio-sidecar-injection` webhook, which modifies the pod or deployment manifest
- Resources are then created based on the modified manifest



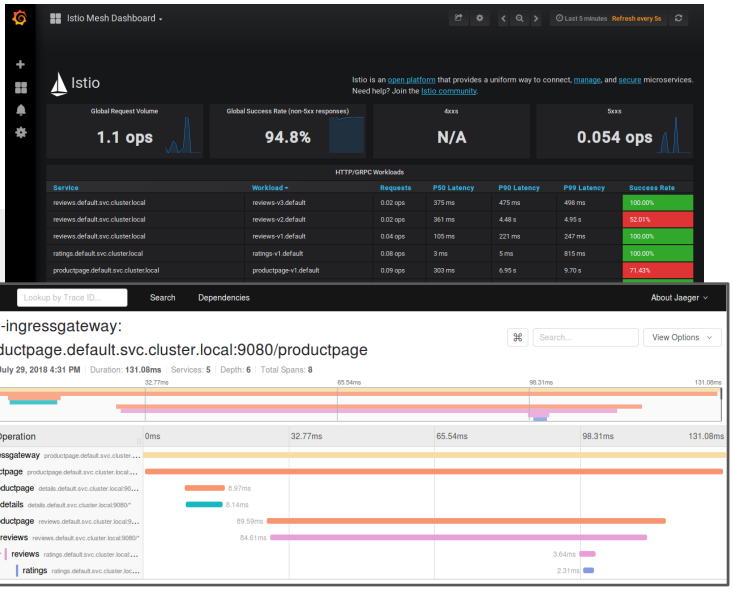
IBM Training

Istio interfaces

- The istioctl CLI
- Istio Grafana dashboards
- Istio Jaeger tracing

```
Usage:
  istioctl [command]

Available Commands:
  authn          Interact with Istio authentication policies
  context-create Create a kubeconfig file suitable for use
  create         Create policies and rules
  delete         Delete policies or rules
  deregister     De-registers a service instance
  experimental   Experimental commands that may be modified
  gen-deploy     Generates the configuration for Istio's c
  get            Retrieve policies and rules
  help           Help about any command
  kube-inject    Inject Envoy sidecar into Kubernetes pod
  proxy-config   Retrieve information about proxy configur
  proxy-status   Retrieves the synchronization status of e
  register       Registers a service instance (e.g. VM) jo
  replace        Replace existing policies and rules
  version        Prints out build version information
```



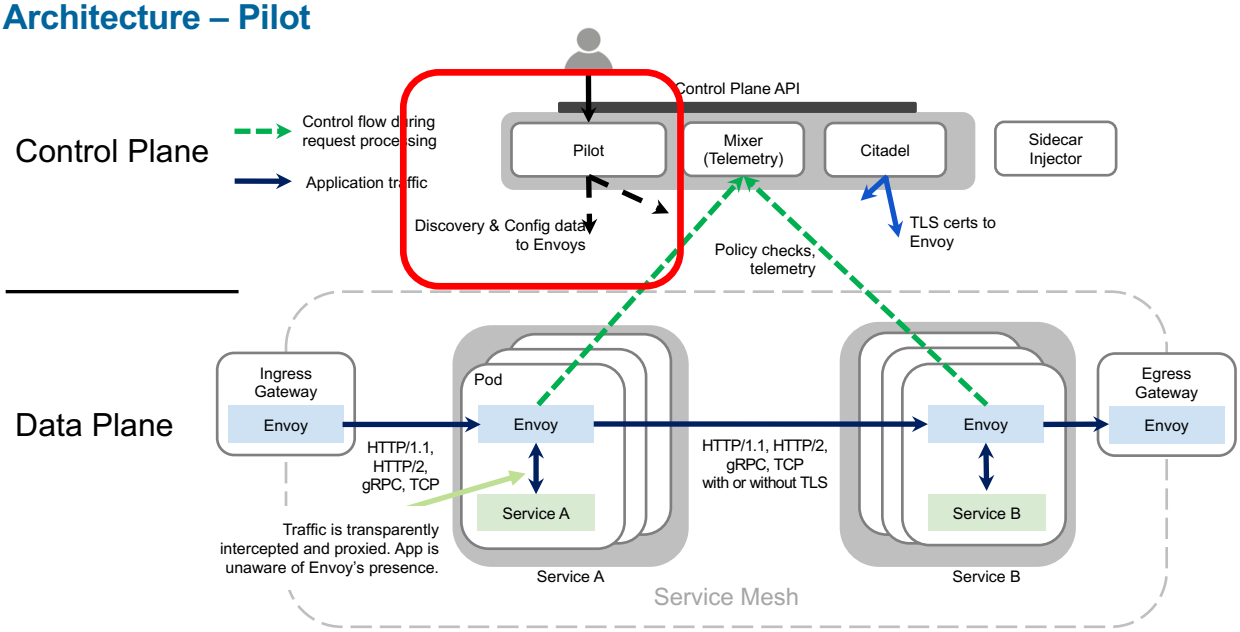
IBM Training

Istio Traffic Management

Routing Rules Configuration

IBM Training

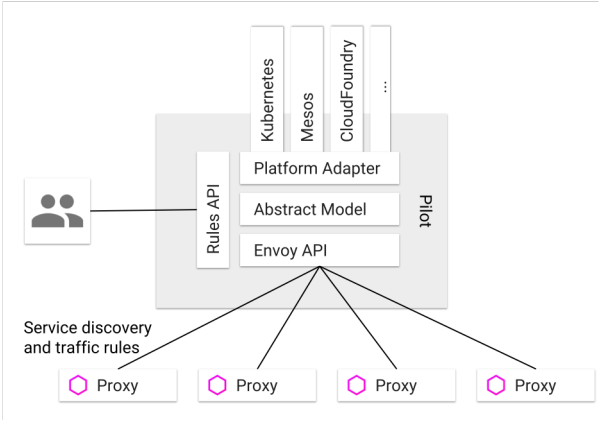
Architecture – Pilot



IBM Training

Traffic Management

- **Pilot**
  - Define the rules you want traffic to follow, rather than which service instances should receive traffic
  - Manages and configures all the Envoy proxy instances deployed in a particular Istio service mesh
  - Maintains a canonical model of all the services in the mesh
  - Uses the model to let Envoy instances know about the other Envoy instances in the mesh via its discovery service
  - Manages the lifecycle of each Envoy instance
- **Envoy**
  - Each instance maintains load balancing information (from Pilot and periodic health-checks)
  - Intelligently distributes traffic between destination instances while following its specified routing rules



IBM Training

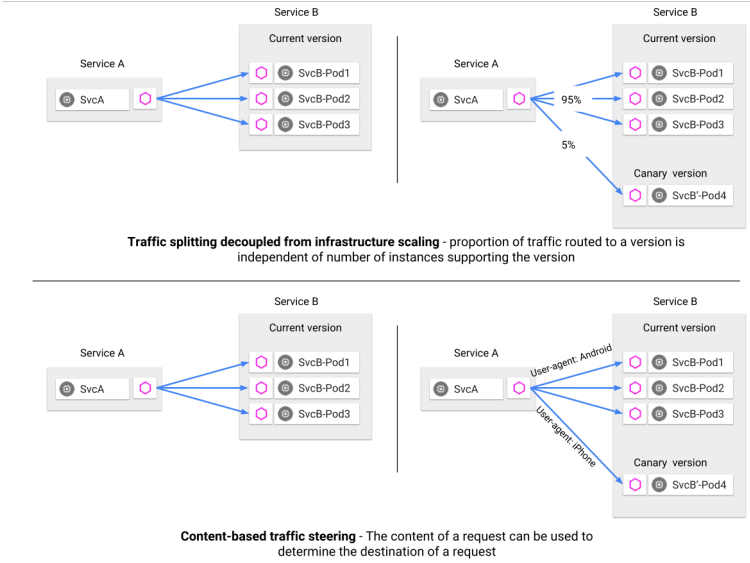
Traffic Routing Terms

Applications address only the destination service without knowledge of individual service subsets or other mappings. The actual choice of the version is determined by Envoy, enabling the application code to decouple itself from the evolution of dependent services.

- **Service**
  - A unit of application behavior bound to a unique name in a service registry (a.k.a. downstream application)
  - Services consist of multiple network endpoints implemented by workload instances running on pods, containers, VMs, etc.
- **Service versions** (a.k.a. **subsets**)
  - Different implementation iterations of the same logical application behavior
  - Instances implement different versions (v1, v2) or are deployed in different environments (dev, stage, prod)
- **Source**
  - The client calling a service
- **Host**
  - The address used by a client (source) when attempting to connect to a service

IBM Training

Routing: Traffic Splitting and Traffic Steering



- Traffic splitting
  - Routing not based on the request content
  - Staged rollouts with %-based traffic splits
- Traffic steering
  - Routing based on the contents of the request
- Either can be used for:
  - Canary rollouts
  - A/B testing
  - Red/black deployment

IBM Training

Rule Configuration

Traffic routing—between a sources and services–is defined as rules

Traffic routing rules are configured using four types of definitions

- **VirtualService**
  - Defines the rules that control how requests for a service are routed to service subsets
- **DestinationRule**
  - Configures the policies to be applied to a request based on the selected service subset
- **ServiceEntry**
  - Commonly used to enable requests to services outside of an Istio service mesh
- **Gateway**
  - Configures a load balancer for HTTP/TCP traffic
  - Most commonly operates at the edge of the mesh to enable ingress or egress traffic for an application

IBM Training

VirtualService definition

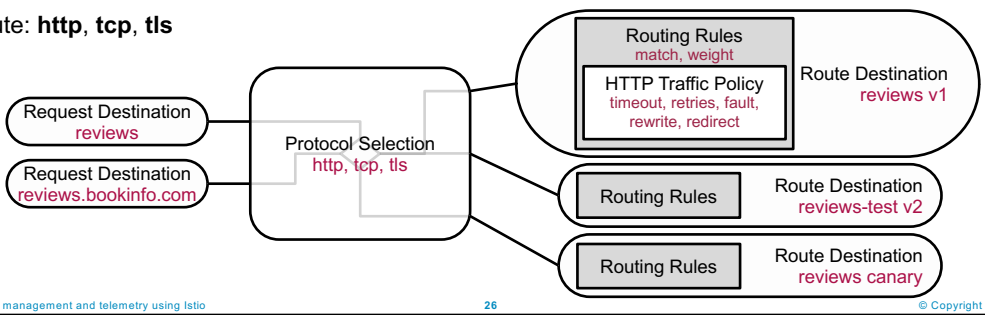
- Rules for routing a service to its service subset
  - Configures how requests should be split across or steered to service subsets
  - Acts as a load-balanced destination from a gateway
- Used to configure
  - Traffic splitting
  - Traffic steering
- For HTTP connections
  - Timeouts and retries
  - Fault injection
  - URL rewriting

```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: reviews
spec:
  hosts:
  - reviews
  http:
  - match:
    - headers:
        end-user:
          exact: jason
    route:
    - destination:
        host: reviews
        subset: v2
  - route:
    - destination:
        host: reviews
        subset: v3
```


IBM Training

VirtualService illustration

- Request destinations
  - Hosts that sources can invoke
  - Attributes: **hosts** and **gateways**
- Route destinations
  - Subset of the destination
  - Attribute: **route** and **destination**
- Protocol selection
  - How to connect to the destination subset
  - Attribute: **http**, **tcp**, **tls**
- Routing rules
  - Additional routing attributes to be applied for the route destinations
  - Attributes: **weight** and **match**
- HTTP traffic policy
  - Protocol-specific connection quality of service
  - Attributes: **timeout**, **retries**, **fault**, **rewrite**, and **redirect**

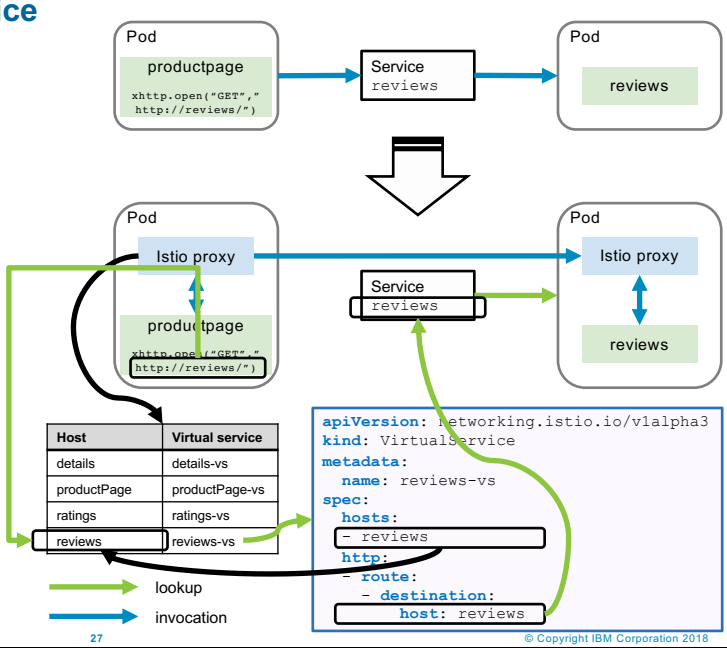


IBM Training

VirtualService – Modifying a service

How a Kubernetes service becomes an Istio virtual service

- Proxy matches request to virtual service
  - Proxy has a table of virtual services
  - Proxy selects the one whose host matches the request
- Virtual service selects the service
  - The virtual service specifies a route to a destination host
  - Proxy finds the service matching the destination host
- Proxy uses service to invoke destination
  - Proxy uses the service to find host pods
  - Proxy load balances across the host pods



IBM Training

DestinationRule definition

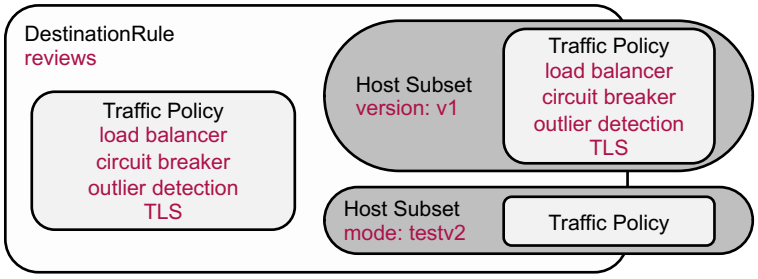
- Defines subsets for a virtual service
- Policies for how to identify and invoke a service subset
  - All for the same host
  - TrafficPolicy can be specified for:
    - all subsets
    - specifically for a subset
    - certain ports for all subsets
    - certain ports within a subset
- Used to configure
  - Load balancer settings
  - Connection pool settings
  - Outlier detection
  - TLS settings

```
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: httpbin
spec:
  host: httpbin
  trafficPolicy:
    connectionPool:
      tcp:
        maxConnections: 10
      http:
        http1MaxPendingRequests: 10
        maxRequestsPerConnection: 5
    outlierDetection:
      consecutiveErrors: 1
      interval: 1s
      baseEjectionTime: 3m
      maxEjectionPercent: 100
  subsets:
  - name: main
    labels:
      version: v23
  - name: new
    labels:
      version: v24
  - name: beta
    labels:
      usage: test
```

IBM Training

DestinationRule illustration

- Destination host
  - Host that route destinations can select
  - Attribute: **host**
- Host subset
  - Identify a subset of service endpoints
  - Attribute: **labels**
- Traffic policy
  - Influence expected quality of service for destinations
  - Attributes: **trafficPolicy**
    - **loadBalancer**, **connectionPool**, **outlierDetection**, and **tls**
- Traffic policy can be applied to a port
  - Attribute: **portLevelSettings**

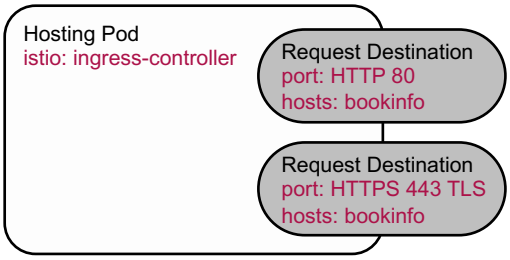


Microservices traffic management and telemetry using Istio 29 © Copyright IBM Corporation 2018

IBM Training

Gateway definition

- Load balancer operating at the edge of the mesh receiving incoming HTTP/TCP connections
  - Configures ports to expose externally
  - Maps each exposed port to a request destination
  - Each gateway can have one or more Virtual Services that defines these request destinations
- Gateway is attached to Istio Ingress Controller (which can be the Kubernetes Ingress Controller)
- Hosting pod
  - Label of pod that will host the gateway
  - Attribute: **selector**
- Request destinations
  - Ports for the gateway to expose and hosts for the corresponding services
  - Attributes: **servers**
    - **port**/**hosts**/**tls** triplet

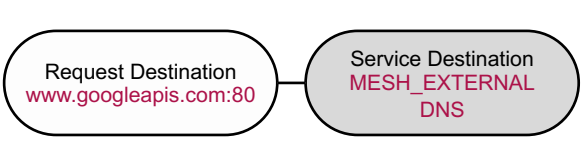


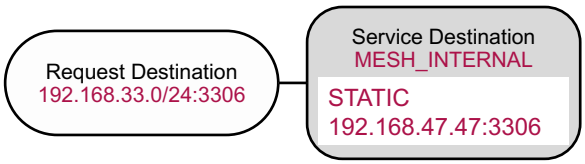
Microservices traffic management and telemetry using Istio 30 © Copyright IBM Corporation 2018

IBM Training

ServiceEntry definition

- Enables adding additional entries into Istio's internal service registry
  - Makes them auto-discoverable within the mesh
  - Route to these services with traffic policies
- Used to configure additional connections
  - Connections to off-premises service endpoints
    - E.g. web APIs
  - Connections to on-premises service endpoints that are not part of the platform's service registry
    - E.g. VMs (instead of containers) hosting services
- Request destination
  - Hosts that sources can invoke
  - Hostname attributes: **hosts** and **ports**
  - IP address attributes: **addresses** and **ports**
- Service destination
  - The existing service's endpoint
    - For DNS resolution, endpoint is **host**:**port**
  - Attributes: **location**, **resolution**, and **endpoints**
  - External: Policy enforced client-side, mTLS disabled; enable mTLS in the Gateway
  - Internal: Policy enforced server-side, mTLS possible





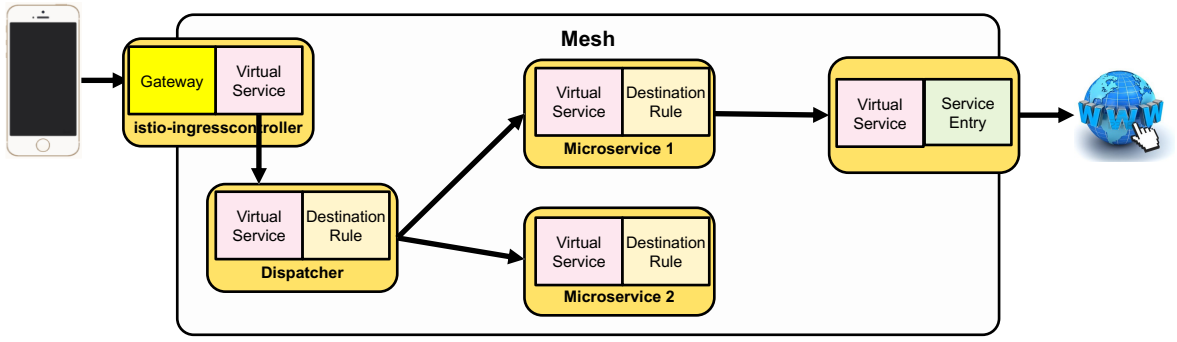
IBM Training

Istio Traffic Management

Traffic Routing

IBM Training

Traffic policy definition for Microservices application

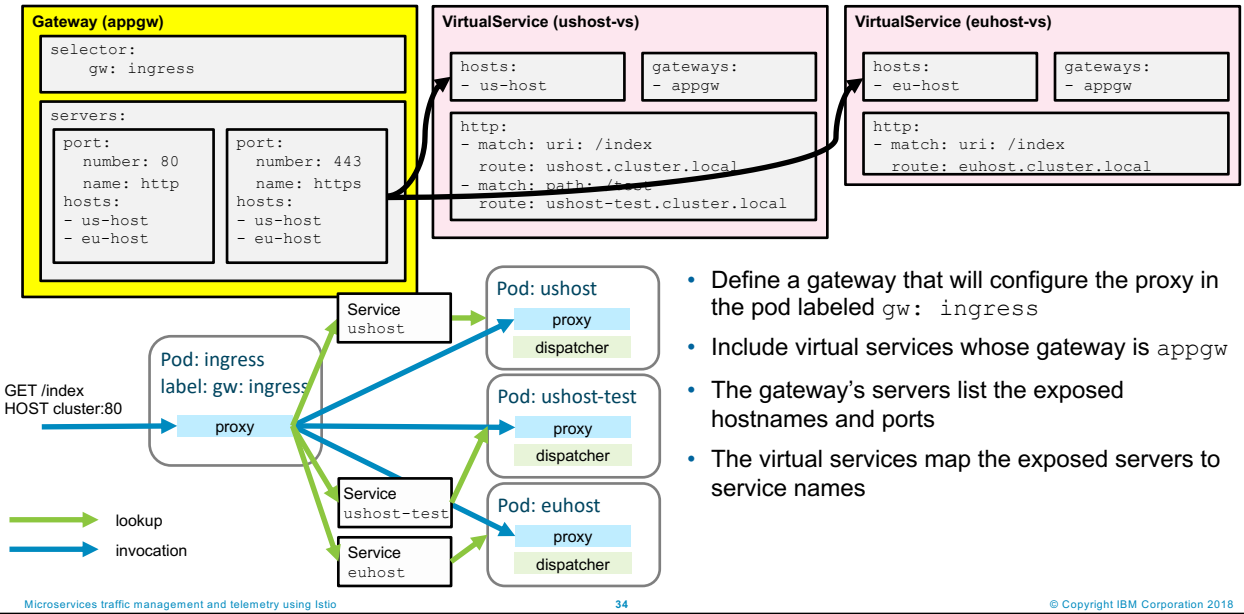


- Client-facing components are registered in the ingress Gateway via a Virtual Service
- Each application component is configured in two parts
  - A Virtual Service defines where to send the request to
  - A Destination Rule specifies traffic control attributes for the specific destination
- Application components access an external service via a Service Entry
  - A Virtual Service makes the Service Entry available as a host in the mesh

Microservices traffic management and telemetry using Istio 33 © Copyright IBM Corporation 2018

IBM Training

Inbound definition

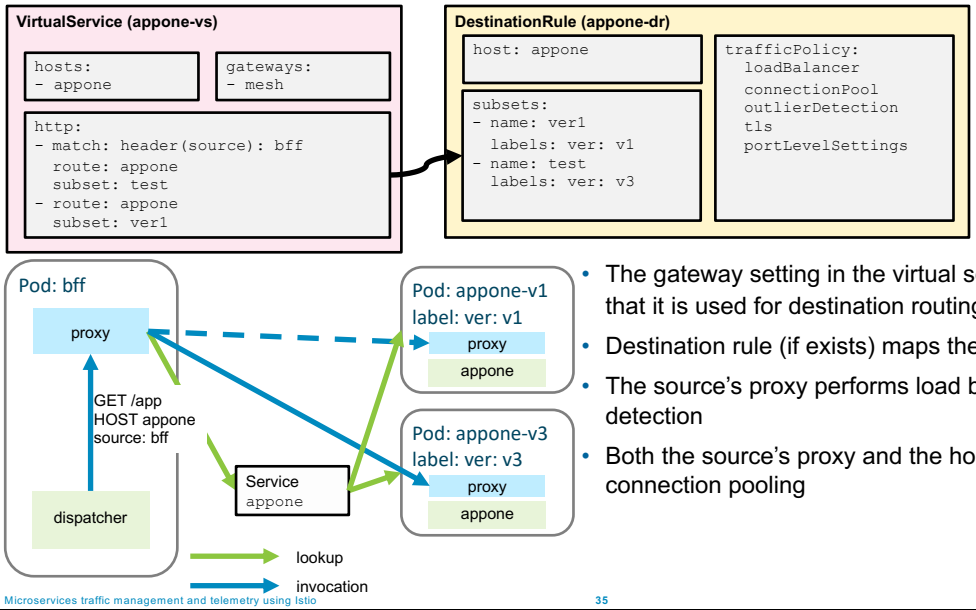


- Define a gateway that will configure the proxy in the pod labeled `gw: ingress`
- Include virtual services whose gateway is `appgw`
- The gateway's servers list the exposed hostnames and ports
- The virtual services map the exposed servers to service names

Microservices traffic management and telemetry using Istio 34 © Copyright IBM Corporation 2018

IBM Training

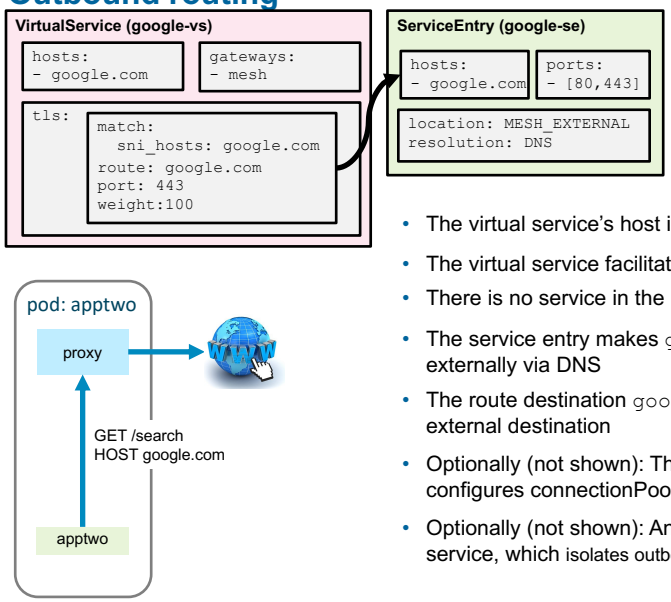
Traffic routing definition for in-mesh destination



- The gateway setting in the virtual service, `mesh`, specifies that it is used for destination routing
- Destination rule (if exists) maps the subsets to pod labels
- The source's proxy performs load balancing and outlier detection
- Both the source's proxy and the host's proxy performs connection pooling

IBM Training

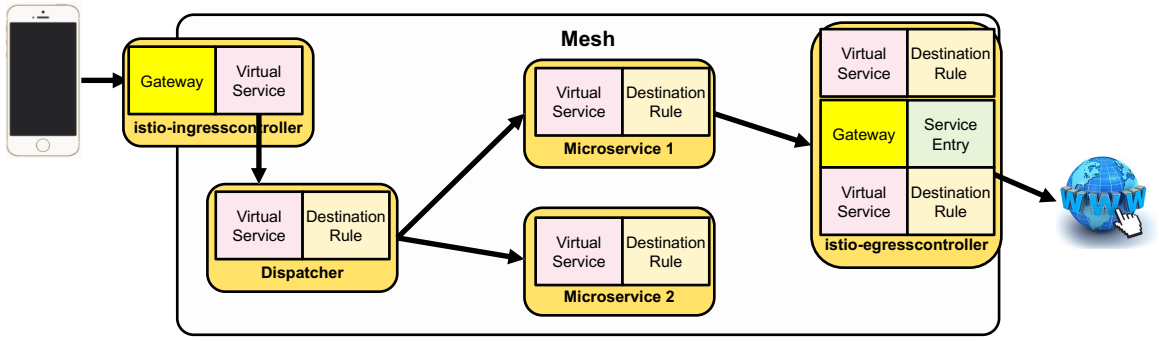
Outbound routing



- The virtual service's host is `google.com`, which is what the client is invoking
- The virtual service facilitates TLS termination using the TLS route rule
- There is no service in the mesh named `google.com`
- The service entry makes `google.com` available as a known host that's available externally via DNS
- The route destination `google.com` therefore routes not to a service but to the external destination
- Optionally (not shown): The virtual service can route to a destination rule that configures connectionPool, outlierDetection, etc.
- Optionally (not shown): An egress gateway can be added using a gateway and virtual service, which isolates outbound traffic and enables it to be measured

IBM Training

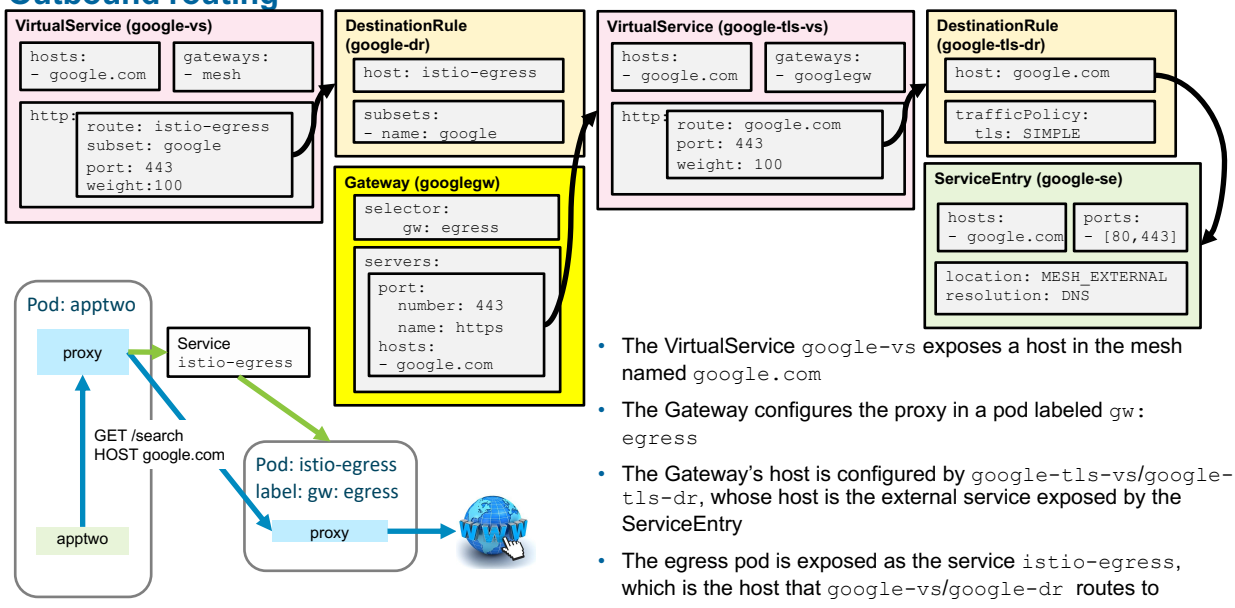
Traffic policy definition for Microservices application



- The Service Entry can be enhanced into a full egress gateway
  - The Gateway will use the Service Entry via a Virtual Service
  - Another Virtual Service is used to expose the Gateway ports as hosts in the mesh

IBM Training

Outbound routing



- The VirtualService `google-vs` exposes a host in the mesh named `google.com`
- The Gateway configures the proxy in a pod labeled `gw: egress`
- The Gateway's host is configured by `google-tls-vs`/`google-tls-dr`, whose host is the external service exposed by the ServiceEntry
- The egress pod is exposed as the service `istio-egress`, which is the host that `google-vs`/`google-dr` routes to

IBM Training

Traffic policy

- Each DestinationRule can configure one or more traffic policies
  - Influence expected quality of service for destinations
  - Better fault tolerance for using the service
  - Attribute: **trafficPolicy**
- Three traffic policy configurations affect resiliency
  - Load balancer (**loadBalancer**)
  - Connection pool (**connectionPool**)
  - Outlier detection (**outlierDetection**)
- While traffic policy configures resiliency, the service proxy implements it
  - Envoy implements load balancing, configured by DestinationRule's load balancing settings
  - Envoy implements circuit breaking, configured by DestinationRule's connection pool settings
  - Envoy implements outlier detection, configured by DestinationRule's outlier detection settings

IBM Training

Traffic policy: Load balancing

- Istio configures load balancing logic
  - Load balancing is performed against the pool of pods that are targeted by the DestinationRule subset
  - The default is simple ROUND_ROBIN load balancing
- Simple load balancing is implemented by Envoy
  - Envoy has more load balancer settings than Istio supports
- Istio simple load balancer settings
  - ROUND_ROBIN
  - LEAST_CONN
  - RANDOM
  - PASSTHROUGH
- Load balancer can be simple or use session affinity
- Session affinity load balancer
  - Host is determined by a hash (**consistentHash**)
- Elements in the session affinity hash
  - httpHeaderName
  - httpCookie
  - useSourceIp
  - minimumRingSize (hash ring, with each node assigned to a number in a ring)

IBM Training

Envoy health checking

- Envoy has passive and active health checking
  - Ideally, active health checking detects and ejects a problematic host before a service request fails
- Active health checking
  - Envoy sends tests to the host, independent of service requests
- Passive health checking
  - Envoy notes when service requests fail
- Passive health checking is configured by outlier detection

IBM Training

Traffic policy: Connection pool and Outlier detection

Connection Pools

- Connection pool
  - Limit the number of requests to a destination
  - Avoids overloading the destination
  - Implemented by Envoy circuit breaker
- Connection pool TCP settings
  - maxConnections
  - connectTimeout
- Connection pool HTTP settings
  - http1MaxPendingRequests
  - http2MaxRequests
  - maxRequestsPerConnection
  - maxRetries

Outlier Detection

- Outlier detection
  - Removes hosts from the load balancer set that are not performing properly
  - Implemented by Envoy outlier detection
- Outlier detection settings
  - consecutiveErrors
  - interval
  - baseEjectionTime
  - maxEjectionPercent

IBM Training

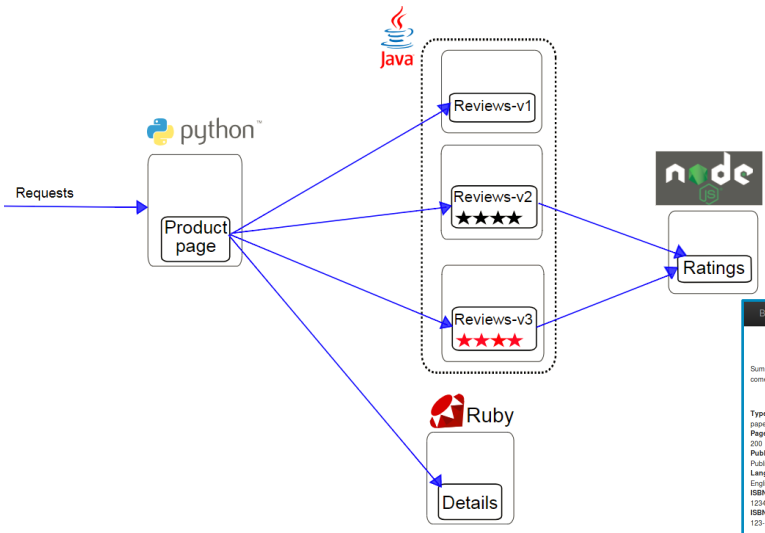
Istio Traffic Management

Bookinfo Example

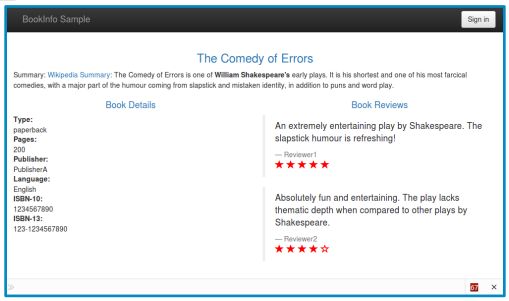
43

IBM Training

The Bookinfo application



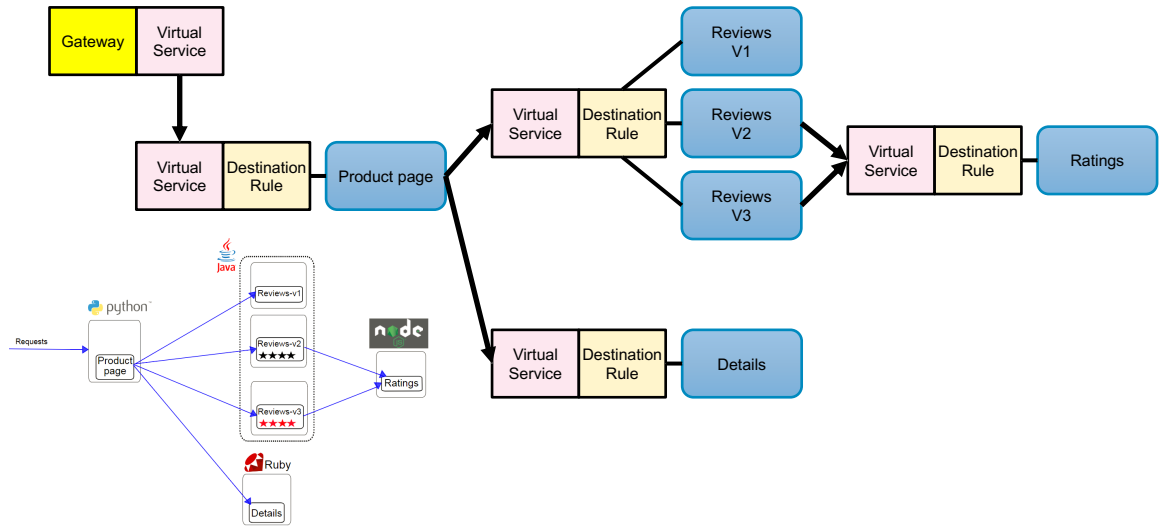
- Polyglot application
  - Three microservices
  - Single front end entry point
- Three versions of Reviews app
  - Allows various tests to be conducted for routing and reliability features



Microservices traffic management and telemetry using Istio 44

IBM Training

Bookinfo istio configuration



IBM Training

Gateway manifest

Gateway | Virtual Service

```
localuser@ibmcloudacademy:~/istio-0.8.0$ kubectl get svc -n istio-system
NAME                       TYPE           CLUSTER-IP     EXTERNAL-IP   PORT(S)                                                                     AGE
istio-citadel              ClusterIP      10.0.0.104     <none>        8060/TCP,9093/TCP                                                           2d
istio-egressgateway        ClusterIP      10.0.0.156     <none>        80/TCP,443/TCP                                                              2d
istio-ingress              LoadBalancer   10.0.0.186     <pending>     80:32000/TCP,443:31968/TCP                                                  2d
istio-ingressgateway       LoadBalancer   10.0.0.249     <pending>     80:31380/TCP,443:31390/TCP,31400:31400/TCP                                  2d
istio-pilot                ClusterIP      10.0.0.141     <none>        15003/TCP,15005/TCP,15007/TCP,15010/TCP,15011/TCP,8080/TCP,9093/TCP         2d
istio-policy               ClusterIP      10.0.0.227     <none>        9091/TCP,15004/TCP,9093/TCP                                                 2d
istio-sidecar-injector     ClusterIP      10.0.0.93      <none>        443/TCP                                                                     2d
istio-statsd-prom-bridge   ClusterIP      10.0.0.125     <none>        9102/TCP,9125/UDP                                                           2d
istio-telemetry            ClusterIP      10.0.0.157     <none>        9091/TCP,15004/TCP,9093/TCP,42422/TCP                                       2d
prometheus                 ClusterIP      10.0.0.227     <none>        9090/TCP                                                                    2d
```

```
Name:         istio-ingressgateway
Namespace:    istio-system
Labels:       chart=ingressgateway-0.8.0
              heritage=Tiller
              istio=ingressgateway
              release=istio
```

```
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: bookinfo-gateway
spec:
  selector:
    istio: ingressgateway # use istio default controller
  servers:
  - port:
      number: 80
      name: http
      protocol: HTTP
    hosts:
    - "*"
```

```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: bookinfo
spec:
  hosts:
  - "*"
  gateways:
  - bookinfo-gateway
  http:
  - match:
    - uri:
        exact: /productpage
    - uri:
        exact: /login
    - uri:
        exact: /logout
    - uri:
        prefix: /api/v1/products
    route:
    - destination:
        host: productpage
        port:
          number: 9080
```

IBM Training

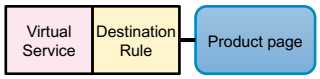
Product page virtual service

- VirtualService is selected based on the source destination
- The route destination links the VirtualService to the DestinationRule
- This simple DestinationRule directly refers to the pod (by finding the appropriate service name)

```
localuser@ibmcloudacademy:~$ kubectl get svc
NAME          TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)    AGE
details       ClusterIP   172.21.178.171   <none>        9080/TCP   3h
kubernetes    ClusterIP   172.21.0.1       <none>        443/TCP    13d
productpage   ClusterIP   172.21.49.178    <none>        9080/TCP   3h
ratings       ClusterIP   172.21.74.214    <none>        9080/TCP   3h
reviews       ClusterIP   172.21.16.141    <none>        9080/TCP   3h
```

```
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: productpage
spec:
  host: productpage
```

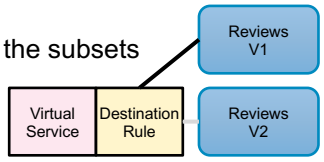
```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: productpage
spec:
  hosts:
  - productpage
  http:
  - route:
    - destination:
        host: productpage
```



IBM Training

Defining subsets using DestinationRule

- With multiple reviews application deployed, a DestinationRule defines the subsets
- Initially the VirtualService would refer to only the original version (version: v1)



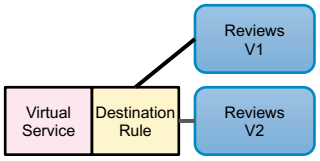

```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: reviews
spec:
  hosts:
  - reviews
  http:
  - route:
    - destination:
        host: reviews
        subset: v1
```

```
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: reviews
spec:
  host: reviews
  subsets:
  - name: v1
    labels:
      version: v1
  - name: v2
    labels:
      version: v2
```

IBM Training

Traffic splitting across multiple destinations

- For Canary testing, a temporary change of the VirtualService would allow some traffic to be passed to v2
- For Blue-green deployment VirtualService rule can be seamlessly updated in intervals to shift workload from v1 to v2



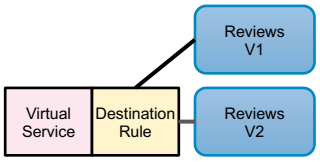

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: reviews
spec:
  hosts:
  - reviews
  http:
  - route:
    - destination:
      host: reviews
      subset: v1
      weight: 90
    - destination:
      host: reviews
      subset: v2
      weight: 10
```

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: reviews
spec:
  host: reviews
  subsets:
  name: v1
  labels:
    version: v1
  name: v2
  labels:
    version: v2
```


IBM Training

Content-based Traffic steering examples

- With traffic steering, depending on the content of the HTTP request, some of the traffic can be directed to a certain set of destination
- Matching can be performed against source labels, URI, headers, port, and other HTTP attributes



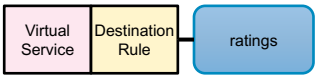

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: reviews
spec:
  hosts:
  - reviews
  http:
  - match:
    - headers:
        end-user:
          exact: jason
    route:
    - destination:
      host: reviews
      subset: v2
  - route:
    - destination:
      host: reviews
      subset: v1
```

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: reviews
spec:
  host: reviews
  subsets:
  name: v1
  labels:
    version: v1
  name: v2
  labels:
    version: v2
```

IBM Training

HTTP fault injection and testing

- Fault injection can be used for testing
  - Faults are caused on a percentage of requests
  - Faults can cause a request delay or failure
- In this example, ratings is being invoked
  - All of the requests from bar have a 2 second timeout
  - 40% of the requests from bar also have a 5 second delay
  - 20% of the requests from foo get an HTTP 500 error
  - All other requests (not foo or bar) have a 4 second timeout
- Timeout is measured after the host is invoked, so timeout is calculated after delay period has passed
  - The 40% of requests from bar that have the delay will time out after 7 seconds (5 sec delay + 2 sec timeout)

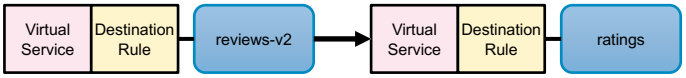



```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: ratings
spec:
  hosts:
  - ratings
  http:
  - match:
    - headers:
        end-user:
          exact: bar
    fault:
      delay:
        percent: 40
        fixedDelay: 5s
    timeout: 2s
    route:
    - destination:
        host: ratings
  - match:
    - headers:
        end-user:
          exact: foo
    fault:
      abort:
        percent: 20
        httpStatus: 500
    route:
    - destination:
        host: ratings
  - route:
    - destination:
        host: ratings
    timeout: 4s
```

Microservices traffic management and telemetry using Istio 51 oration 2018

IBM Training

Testing circuit breaker



- Connection pool
  - Limits connections for reviews to invoke ratings
  - Limited to 1 concurrent connection, 1 request per connection
    - One concurrent request total
  - While requests are using all of the connections in a pool, any new requests are rejected
- Outlier detection
  - If there are 3 requests in 2 seconds, ratings will be ejected for 3 minutes
  - Request 1 will take more than 2 seconds, blocking the connection during that time
  - Request 2 won't get a connection, which will generate the first error
  - Request 3 won't get a connection, which will generate the second error, causing ejection
- Fixed delay
  - ratings is delayed 2 seconds before the service runs
  - Guarantees that the service will run for more than 2 seconds

```
apiVersion: ...
kind: DestinationRule
metadata:
  name: reviews-dr
spec:
  host: reviews
  trafficPolicy:
    connectionPool:
      tcp:
        maxConnections: 1
      http:
        http1MaxPendingRequests: 1
        maxRequestsPerConnection: 1
    outlierDetection:
      consecutiveErrors: 2
      interval: 2s
      baseEjectionTime: 3m
      maxEjectionPercent: 100
```

```
apiVersion: ...
kind: VirtualService
metadata:
  name: ratings
spec:
  hosts:
  - ratings
  http:
  - fault:
      delay:
        percent: 100
        fixedDelay: 2s
    route:
    - destination:
        host: ratings
        subset: v1
```

Microservices traffic management and telemetry using Istio 52 © Copyright IBM Corporation 2018

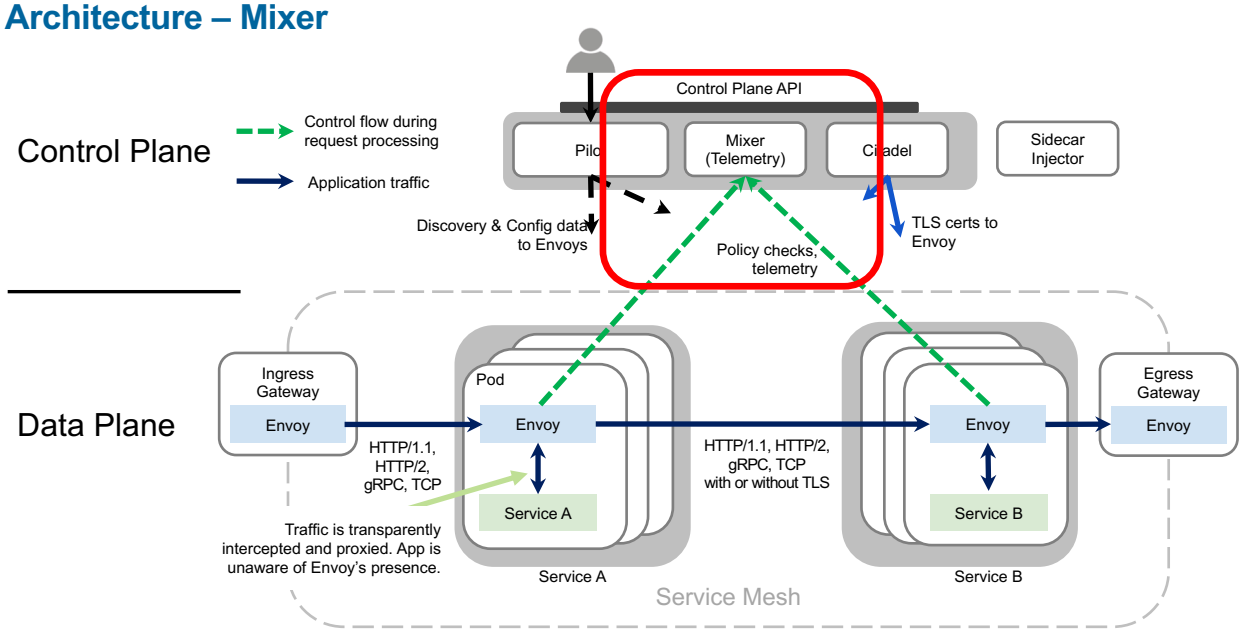
IBM Training

Istio Policies and Telemetry

Mixer Configuration

IBM Training

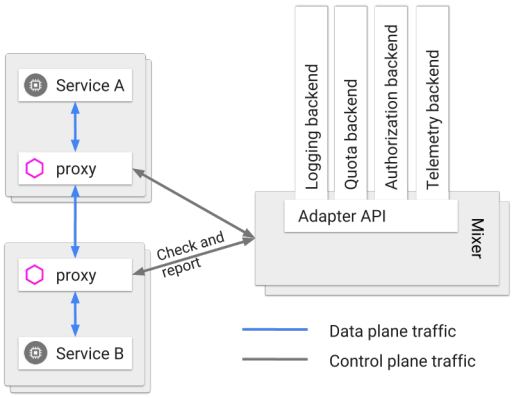
Architecture – Mixer



IBM Training

Policies and Telemetry

- **Mixer**
  - The Istio component responsible for providing policy controls and telemetry collection
  - Enforces authorization policies and collects telemetry for the services in a mesh
  - Adapter API enables it to interface with an open-ended set of infrastructure backends
  - Optional: All policy enforcement and telemetry collection can be disabled
- **Envoy**
  - Performs precondition checks before each request
  - Reports telemetry after each request
  - Calls to Mixer are infrequent
    - Caching and buffering improves performance



Microservices traffic management and telemetry using Istio 55

IBM Training

Attributes

- Each piece of telemetry data is an attribute
- Each attribute has a name and a type
- Mixer is in essence an attribute processing machine
- Each infrastructure backend collects the attributes it's interested in

Attribute examples	
request.path	xyz/abc
request.size	234
request.time	12:34:56.789 04/17/2017
source.ip	192.168.0.1
destination.service	example

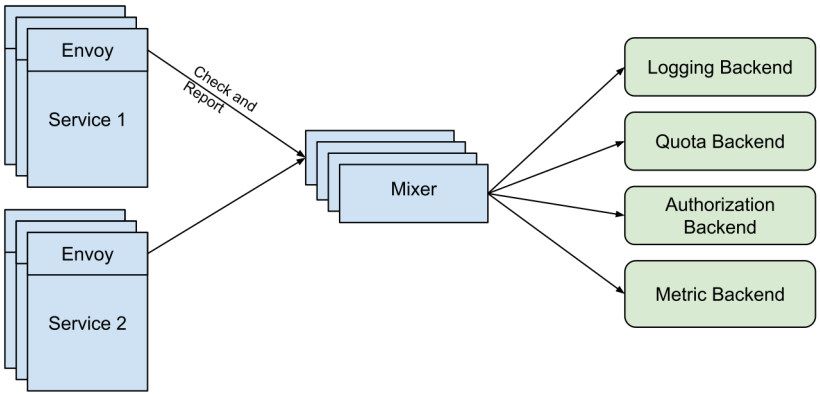
Name	Type	Kubernetes
source.uid	string	kubernetes://…
source.labels	map[string, string]	version => v1
destination.workload.name	string	istio-telemetry
request.headers	map[string, string]	
response.duration	duration	

Microservices traffic management and telemetry using Istio 56

IBM Training

Mixer configuration definitions

- Istio Mixer offers a pluggable interface for the mesh to interact with infrastructure backends to provide:
  - Logging
  - Quota processing
  - Authorization backend
  - Metric metric
- Mixer uses rules to match adapter with template



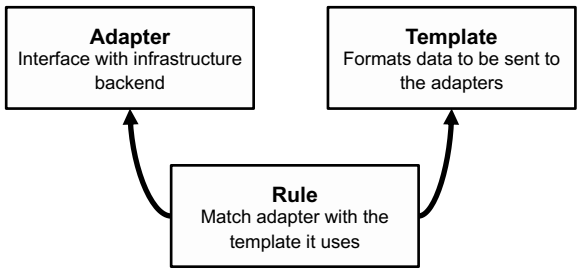
Microservices traffic management and telemetry using Istio 57

IBM Training

Mixer CRD – custom resource definition (templates and adapters)

Templates

Adapters	API Key	Analytics	Authorization	Check Nothing	Kubernetes	List Entry	Log Entry	Metric	Quota	Report Nothing	Service Control Report	Trace Span
Apigee		Y	Y									
Circonus								Y				
CloudWatch (AWS)								Y				
Datadog								Y				
Denier				Y		Y			Y			
Fluentd							Y					
KubernetesEnv					Y							
List						Y						
MemoryQuota									Y			
OPA (Open Policy Agent)			Y									
Prometheus								Y				
RBAC			Y									
Redis Quota									Y			
Service Control	Y								Y		Y	
SignalFx								Y				
SolarWinds							Y	Y				
Stackdriver							Y	Y				Y
StatsD								Y				
Stdio							Y	Y				



Y = Rules that can be written

Y = Rules included in IBM Istio

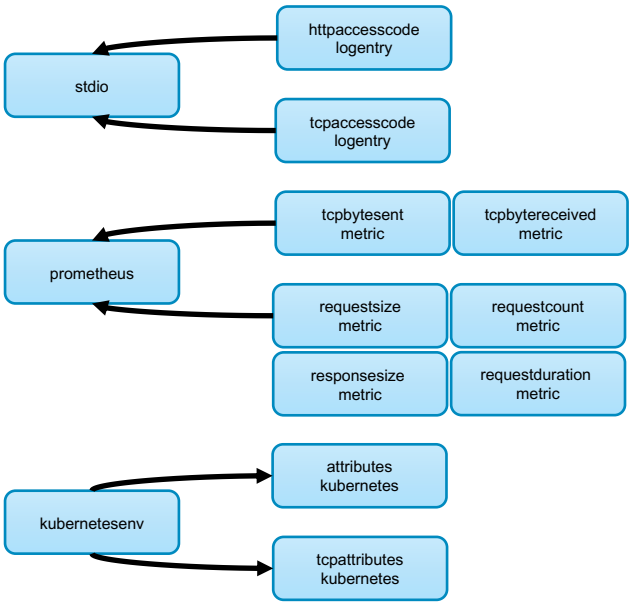
https://istio.io/docs/reference/config/policy-and-telemetry/

Microservices traffic management and telemetry using Istio 58

IBM Training

OOTB definitions

- Output logging entry to telemetry's STDOUT
- Save services metrices to Prometheus
- Collect Kubernetes attributes for other adapters (note the arrow direction)
- Additional definition can be created as needed




```
localuser@tbmcloudacademy:~$ kubectl get rule --all-namespaces
NAMESPACE      NAME                      AGE
istio-system   kubeattrgenrulerule       1d
istio-system   promhttp                  1d
istio-system   promtcp                   1d
istio-system   stdio                     1d
istio-system   stdiotcp                  1d
istio-system   tcpkubeattrgenrulerule    1d
```

Microservices traffic management and telemetry using Istio 59 © Copyright IBM Corporation 2018

IBM Training

Loading metric data to Prometheus

Metric data collected by Mixer, enriched using Kubernetes information and then sent to Prometheus

```
apiVersion: config.istio.io/v1alpha2
kind: metric
metadata:
  name: requestcount
  namespace: istio-system
spec:
  dimensions:
    connection_security_policy: "inbound" | "outbound" | "unknown" | "mutual_tls" | "none"
    destination_app: destination.labels["app"] | "unknown"
    destination_principal: destination.principal | "unknown"
    reporter: "inbound" | "outbound" | "source" | "destination"
    request_protocol: api.protocol | context.protocol | "unknown"
    response_code: response.code | 200
    source_app: source.labels["app"] | "unknown"
    source_principal: source.principal | "unknown"
  monitored_resource_type: '"UNSPECIFIED"'
  value: "1"
```

```
apiVersion: config.istio.io/v1alpha2
kind: prometheus
metadata:
  name: handler
  namespace: istio-system
spec:
  metrics:
  - instance_name: requestcount.metric.istio
    kind: COUNTER
    label_names:
    - reporter
    - source_app
    - source_principal
    - destination_app
    - destination_principal
    - request_protocol
    - response_code
    - connection_security_policy
    name: requests_total
```

```
apiVersion: config.istio.io/v1alpha2
kind: kubernetes
metadata:
  name: attributes
  namespace: istio-system
spec:
  attribute_bindings:
    destination.container.name: $out.destination_container_name | "unknown"
    destination.name: $out.destination_pod_name | "unknown"
    destination.serviceAccount: $out.destination_service_account_name | "unknown"
    source.name: $out.source_pod_name | "unknown"
    source.serviceAccount: $out.source_service_account_name | "unknown"
```

Microservices traffic management and telemetry using Istio 60 © Copyright IBM Corporation 2018

IBM Training

Mixer configuration – Block a service

```
apiVersion: "config.istio.io/v1alpha2"
kind: denier
metadata:
  name: handler
  namespace: istio-system
spec:
  status:
    code: 7
    message: Not allowed
```

```
apiVersion: "config.istio.io/v1alpha2"
kind: checknothing
metadata:
  name: denyrequest
  namespace: istio-system
spec:
```

```
apiVersion: "config.istio.io/v1alpha2"
kind: rule
metadata:
  name: denyingress
  namespace: istio-system
spec:
  match: source.labels["istio"] == "ingressgateway" && request.headers["x-user"] == "john"
  actions:
  - handler: handler.denier.istio-system
    instances: [ denyrequest.checknothing.istio-system ]
```


IBM Training

Istio Debugging

IBM Training

Debugging Istio

- When you have istio running, you may want to:
  - Make sure istio is running well
  - Understand how istio policy affected your application
- Debugging is performed on the control plane (Pilot or Mixer) or on the data plane (Envoy)
  - Control Plane components have an integrated controlZ interface
  - Envoy proxy has opened port 15000 for agent management

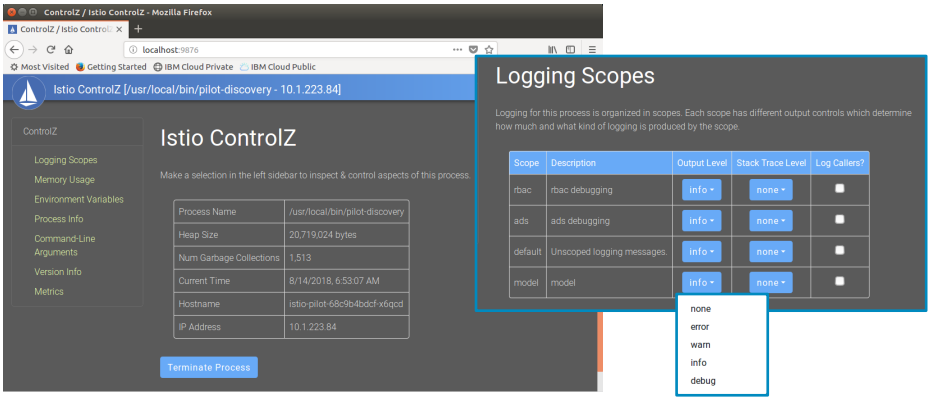
IBM Training

Debugging Istio Pilot/Policy/Telemetry

- Configuring ControlZ
  - Port forwarding controlZ port
  - Accessing controlZ configuration page
- Getting pod log
  - kubectl logs -f <podname> -n istio-system

```
localuser@ibmcloudacademy:~$ kubectl get pod -n istio-system
NAME                                      READY     STATUS    RESTARTS   AGE
grafana-64b7b844cc-vlmkh                  1/1       Running   0          9h
istio-citadel-c8fb4f667-q48lr             1/1       Running   0          9h
istio-egressgateway-6bd595895b-57frv      0/1       Running   0          40s
istio-egressgateway-6bd595895b-8j46f      1/1       Running   0          9h
istio-egressgateway-6bd595895b-msm57      1/1       Running   0          52s
istio-egressgateway-6bd595895b-w78s9      1/1       Running   0          5m
istio-galley-57b749c55d-6kzx2             1/1       Running   0          9h
istio-ingressgateway-57d5878cc9-4wz8h     0/1       Running   0          38s
istio-ingressgateway-57d5878cc9-jhnpb     1/1       Running   0          9h
istio-policy-5465f45b49-8nj5n             2/2       Running   0          1m
```

```
kubectl port-forward istio-pilot-68c9b4bdcf-x6qcd 9876:9876 -n istio-system
```



IBM Training

Envoy configuration

- Getting envoy configuration (JSON)
  `kubectl exec  <podname> -c istio-proxy -- curl localhost:15000/config_dump -s`

```
localuser@ibmcloudacademy:~$ kubectl get pod
NAME                              READY     STATUS    RESTARTS   AGE
details-v1-558d5fc956-7vj9g       2/2       Running   0          8h
productpage-v1-576c55ddb8-mxscj   2/2       Running   0          8h
ratings-v1-556c44f648-2w2h6       2/2       Running   0          8h
reviews-v1-5ff97b656b-tdrcb       2/2       Running   0          8h
reviews-v2-948df8f54-hn2qk        2/2       Running   0          8h
reviews-v3-f9b6c94f8-cflrn        2/2       Running   0          8h
localuser@ibmcloudacademy:~$ kubectl exec details-v1-558d5fc956-7vj9g -c istio-proxy -- curl localhost:15000/config_dump -s
{
 "configs": {
  "bootstrap": {
   "@type": "type.googleapis.com/envoy.admin.v2alpha.BootstrapConfigDump",
   "bootstrap": {
    "node": {
     "id": "sidecar~10.1.235.131~details-v1-558d5fc956-7vj9g.default~default.svc.cluster.local",
     "cluster": "details",
     "metadata": {
      "ISTIO_PROXY_VERSION": "1.0.0",
      "ISTIO_PROXY_SHA": "istio-proxy:6166ae7ebac7f630206b2fe4e6767516bf198313",
      "ISTIO_VERSION": "1.0.0",
      "POD_NAME": "details-v1-558d5fc956-7vj9g",
      "istio": "sidecar",
      "INTERCEPTION_MODE": "REDIRECT"
     },
     "build_version": "0/1.8.0-dev//RELEASE"
    },
    "static_resources": {
     "clusters": [
      {
       "name": "xds-grpc",
       "type": "STRICT_DNS",
       "connect_timeout": "10s",
       "hosts": [
        {
```


IBM Training

Envoy logging and other agent capabilities

- Changing Envoy logging level
  `kubectl exec  <podname> -c istio-proxy –`
  `curl localhost:15000/logging?<name>=<level> -s`

```
localuser@ibmcloudacademy:~$ kubectl exec details-v1-558d5fc956-7vj9g -c istio-proxy -- curl localhost:15000/logging -s
usage: /logging?<name>=<level> (change single level)
usage: /logging?level=<level> (change all levels)
levels: trace debug info warning error critical off
active loggers:
  admin: info
  assert: info
  backtrace: info
  client: info
  config: info
  connection: info
  file: info
  filter: info
  grpc: info
  hc: info
  health_checker: info
  http: info
  http2: info
  hystrix: info
  lua: info
  main: info
  misc: info
  mongo: info
  pool: info
  rbac: info
  redis: info
  router: info
  runtime: info
  stats: info
  testing: info
  thrift: info
  tracing: info
  upstream: info
```

Command	Description
/	Admin home page
/certs	print certs on machine
/clusters	upstream cluster status
/config_dump	dump current Envoy configs (experimental)
/cpuprofiler	enable/disable the CPU profiler
/healthcheck/fail	cause the server to fail health checks
/healthcheck/ok	cause the server to pass health checks
/help	print out list of admin commands
/hot_restart_version	print the hot restart compatibility version
/listeners	print listener addresses
/logging	query/change logging levels
/quitquitquit	exit the server
/reset_counters	reset all counters to zero
/runtime	print runtime values
/runtime_modify	modify runtime values
/server_info	print server version/status information
/stats	print server stats
/stats/prometheus	print server stats in prometheus format

IBM Training

Conclusion

- Concepts
- Traffic management
  - Rule configuration
  - Traffic routing
  - Bookinfo example
- Policies and telemetry
- Debugging Istio

IBM Training

Microservices Network Security

Using Kubernetes and Istio to secure the network in your microservice application

IBM Training

Objectives

- Understand network security options
- Understand Kubernetes network policy basics
- Understand network security with Istio

Prerequisites: Kubernetes, Microservices

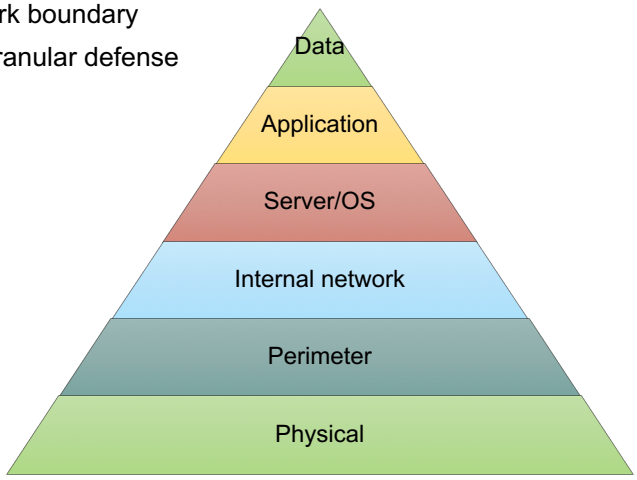
IBM Training

Network security for microservices application

IBM Training

Network security approaches

- Defense in depth
- Layered defense strategy, includes all aspects of IT
- Perimeter protection to stop threat at the network boundary
- Network segmentation and isolation for more granular defense

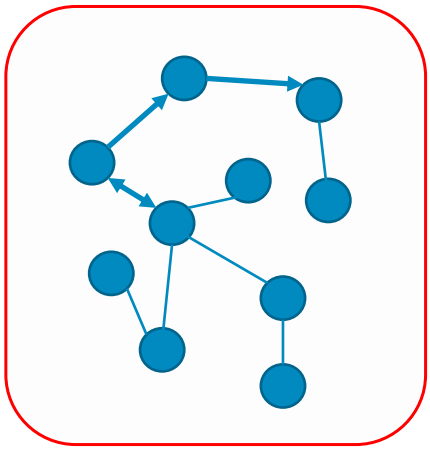


IBM Training

Network perimeter defense

Typical perimeter controls:

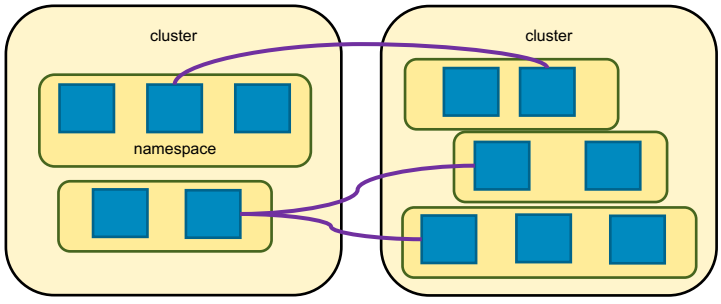
- Firewall and VPN to protect system boundary
- Whitelisting: Deny all by exception, open only what is required
- Intrusion monitoring and protection at system boundary



IBM Training

Network segmentation

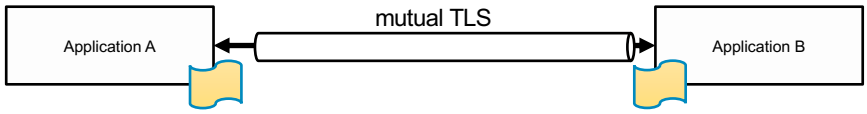
- Segment the network: provides isolation
- Isolation: control what talks to what
- Use segmentation when defining network policies
- Segmentation can be done using namespaces



IBM Training

Communication confidentiality

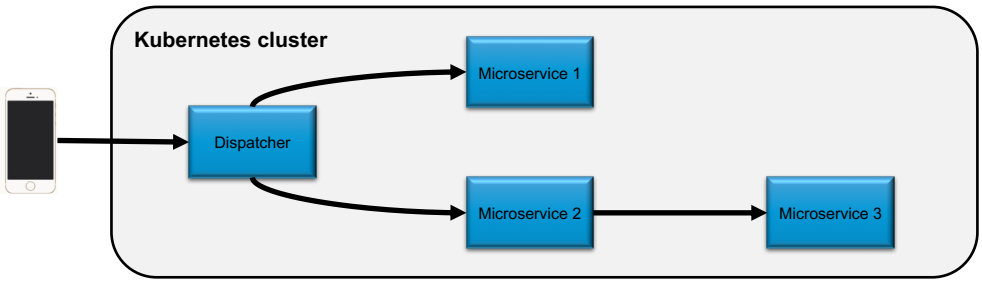
- Traditional TLS issues in a microservices application
  - Applications are not tied to certain hostname or IP addresses
  - Application instances can be created or removed elastically
- Typical TLS certificate scheme are based on identifying hostname or IP address of the involved parties



- Microservices application requires that application identity should be the same regardless of which instances the communication occurs to/from

IBM Training

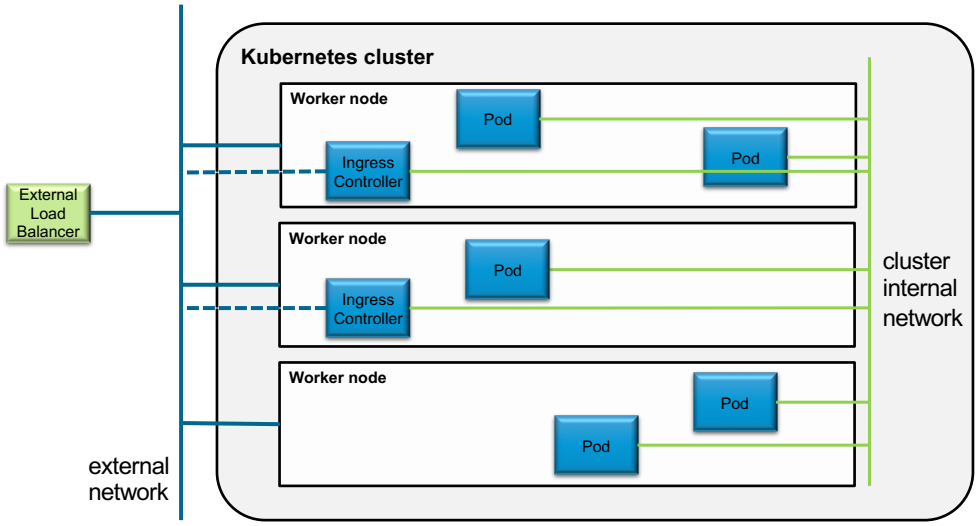
Microservices application in Kubernetes



- How the client can connect to the dispatcher?
- How to ensure that only the dispatcher can invoke Microservice 1 and 2?
- How to restrict the dispatcher from accessing Microservice 3 directly?
- How to prohibit other agents from invoking these microservices apart from the designated ‘callers’?
- **Must understand Kubernetes networking implementation**

IBM Training

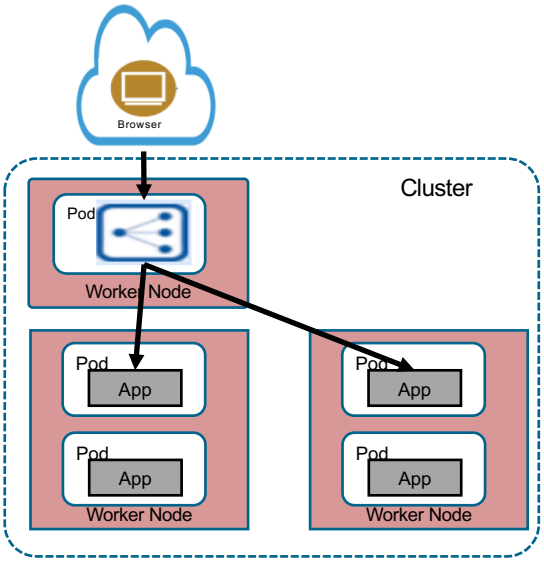
Kubernetes cluster networking



IBM Training

Kubernetes overview

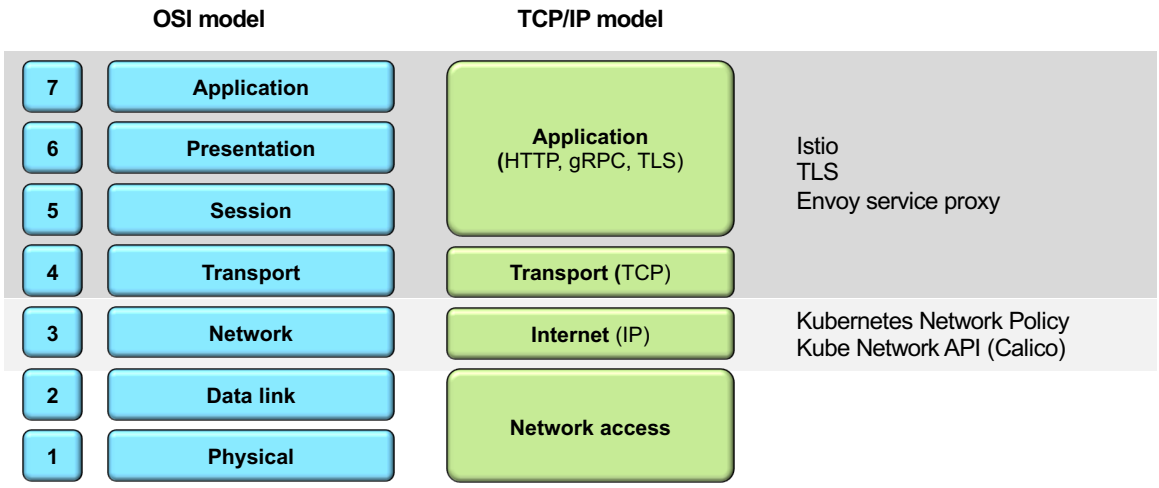
- Container orchestrator
  - Manages a cluster of worker nodes
  - Worker nodes can be virtual machines (VMs) or physical machines
- Apps are deployed as containers in pods
  - Pods run in worker nodes
  - Each pod has a separate IP address
- Apps are exposed as services
  - Each service name is registered in Kube DNS
- Kube Network Policy describes network configuration
  - Policy configures network via Kube Network API
- As pods are created and removed elastically, the cluster uses the API to configure the IPs in the network according to the policy



Annotate and segment with **Labels** and **Namespaces**

IBM Training

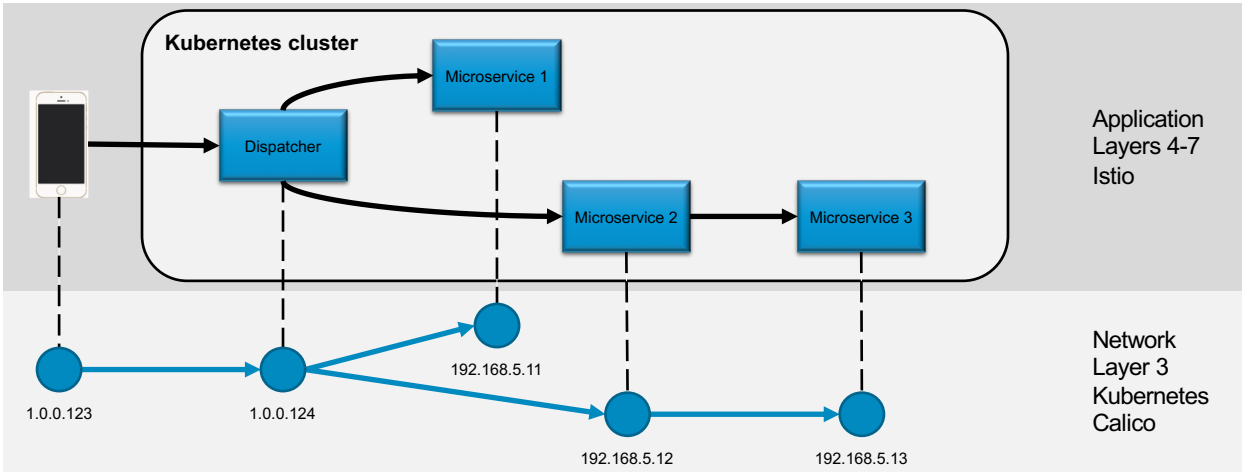
Networking layers – OSI and TCP/IP models



IBM Training

Two layers of application protection

- Configure Kubernetes Network Policy (layer 3) and Istio (layers 4-7) together and consistently
- If Service A can call Service B, define that in both layers using both technologies

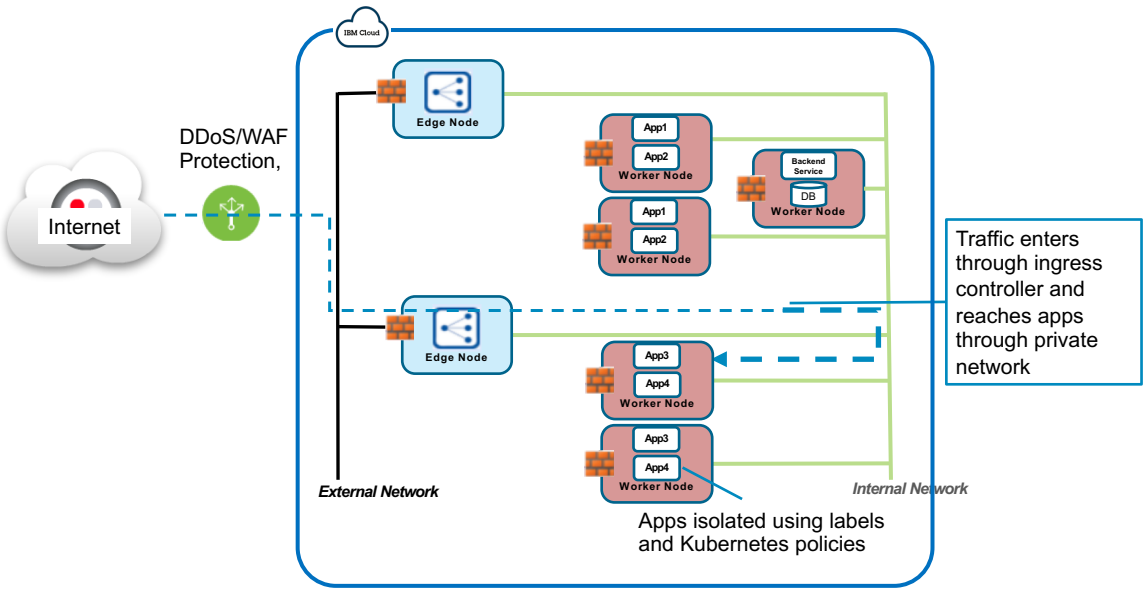


IBM Training

Kubernetes Network Policy

IBM Training

Kubernetes Secure Reference Architecture



IBM Training

The Kubernetes Network Policy resource

- The NetworkPolicy extension is used to define private networking connectivity rules
  - Specifies which pods can communicate and how
- Cluster uses Kubernetes Network API to apply the policy to the network
- IBM uses Calico as its Kubernetes network
  - Calico is a software defined network (SDN) provider
  - Leverages Linux routing and iptables capabilities
- NetworkPolicy defines:
  - Which pod or pods the policy applies to
  - Which resources can connect to the pod (ingress)
  - Which resources the pod can connect to (egress)

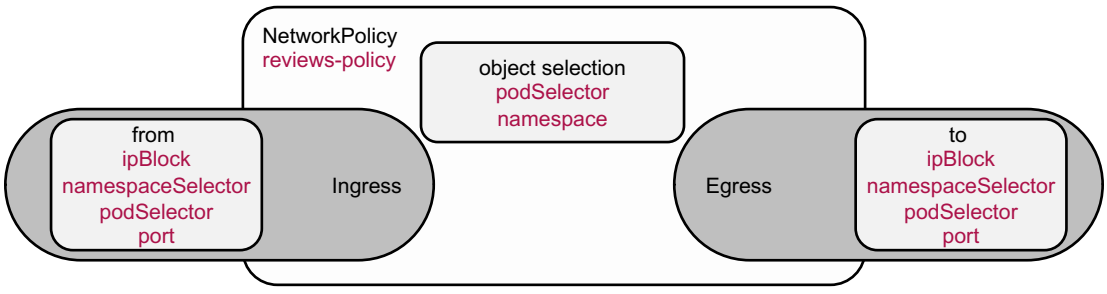
```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: test-network-policy
  namespace: default
spec:
  podSelector:
    matchLabels:
      role: db
  ingress:
  - from:
    - ipBlock:
        cidr: 172.17.0.0/16
        except:
        - 172.17.1.0/24
    - namespaceSelector:
        matchLabels:
          project: myproject
    - podSelector:
        matchLabels:
          role: frontend
    ports:
    - protocol: TCP
      port: 6379
  egress:
  - to:
    - ipBlock:
        cidr: 10.0.0.0/24
    ports:
    - protocol: TCP
      port: 5978
```

Microservices Network Security 15

IBM Training

NetworkPolicy illustration

- Object selection
  - Specifies the resources that this policy applies to
  - Attribute: **podSelector**, **namespace**
- Inbound and Outbound communication filters
  - Identify which communication is being filtered
  - Attribute: **ingress** or **egress**
- Source or target selection
  - Ingress: Access to the selected object
  - Egress: What the selected object can access
  - Attributes: **podSelector**, **namespaceSelector**, **ipBlock**
- Port selection
  - Lists the ports that are allowed access
  - Attribute: **port**



Microservices Network Security 16

IBM Training

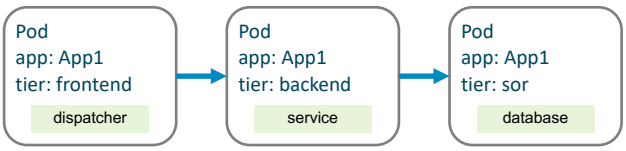
Kubernetes network policy caveats

- Default: Allow all pods to talk to all other pods
  - Allow all traffic to and from a pod if no Kubernetes policy against it
- IBM Cloud Kubernetes Service provides a default set of Calico policies on new clusters
  - allow-node-port-dnat – Allows incoming NodePort, LoadBalancer, and Ingress service traffic to the pods that those services are exposing
  - allow-sys-mgmt – Allows incoming connections for specific IBM Cloud infrastructure (SoftLayer) systems that are used to manage the worker nodes
- Rules are OR'ed, i.e. if any ingress/egress rule matches, connection is allowed
- Connections are duplex: If connection A allowed to B, B can respond but can't create connection to A
- Network policies are scoped to the namespace they are deployed to
- Network Policy can limit both ingress and egress, but limiting ingress only many be sufficient
  - Istio Service Role limits ingress
  - Network Policy egress is still needed for accessing external resources

IBM Training

Network Policy example #1

- Limit which pods can connect on the network
- Two network policies
  - Frontend can access backend
  - Backend can access sor
- Policy uses labels to find pods




```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: backend-allow
spec:
  podSelector:
    matchLabels:
      app: App1
      tier: backend
  ingress:
  - from:
      - podSelector:
          matchLabels:
            app: App1
            tier: frontend
```

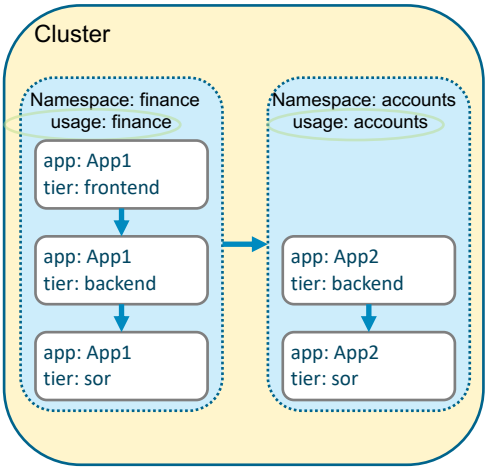
```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: sor-allow
spec:
  podSelector:
    matchLabels:
      app: App1
      tier: sor
  ingress:
  - from:
      - podSelector:
          matchLabels:
            app: App1
            tier: backend
```

IBM Training

Network Policy example #2

- One team's services need to call another's
- One network policy
  - Enables all finance services to call all accounts services on specified port
- Policy uses labels to find namespaces

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  namespace: accounts
  name: accounts-allow
spec:
  podSelector:
  ingress:
  - from:
      - namespaceSelector:
          matchLabels:
            usage: finance
    ports:
      port: 3111
```

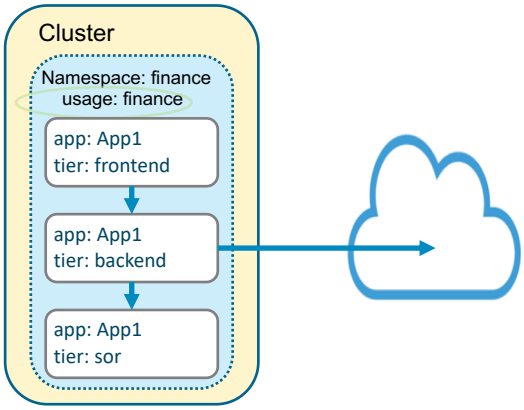


Microservices Network Security 19

IBM Training

Network Policy example #3

- Limit which pods can connect to external web
- Two network policies
  - Block all egress from namespace
  - Enable backend external access on certain ports




```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  namespace: finance
  name: finance-deny-egress
spec:
  podSelector:
  egress: []
```

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  namespace: finance
  name: finance-allow-backend
spec:
  podSelector:
    matchLabels:
      tier: backend
  egress:
  - ports:
    - port: 443
    - port: 53
      protocol: UDP
```

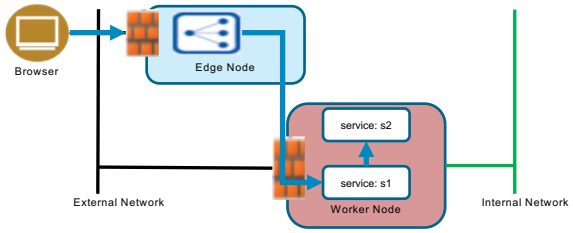
Microservices Network Security 20

IBM Training

Setting up TLS from Ingress

```
apiVersion: v1
kind: Secret
metadata:
  name: tlsSecret
  namespace: default
type: Opaque
data:
  tls.crt: base64 encoded cert
  tls.key: base64 encoded key
```

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: ingress-with-tls
  annotations:
    nginx/ssl-service: s1
spec:
  tls:
  - secretName: tlsSecret
  backend:
    serviceName: s1
    servicePort: 443
```



Encrypt all traffic end to end; TLS 1.2 mandated

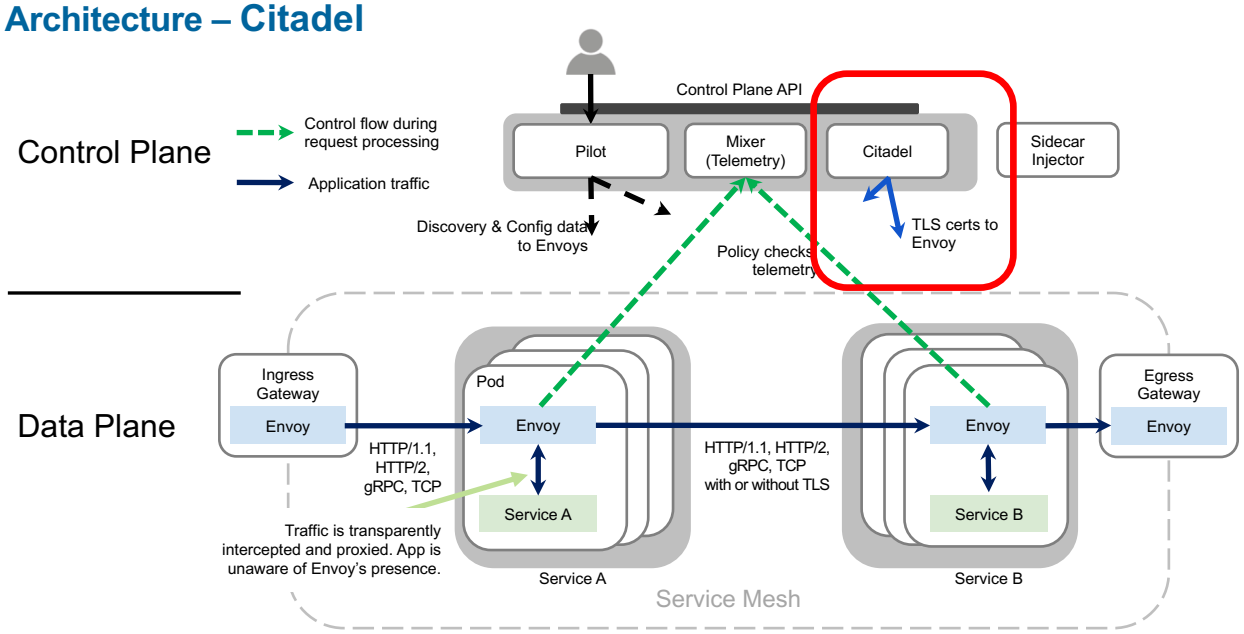
- Enforce TLS at Ingress
- Enforce TLS from Ingress to containers using Ingress 'ssl-service' annotation
- Enforce TLS from component to component
  - Self-signed CA is fine within deployment or cluster across private network
  - Certificates are stored using secrets and managed externally
- In IBM Cloud Public: CA is Digicert (LetsEncrypt in future)

IBM Training

Network Security with Istio

IBM Training

Architecture – Citadel



Microservices Network Security 23 © Copyright IBM Corporation 2018

IBM Training

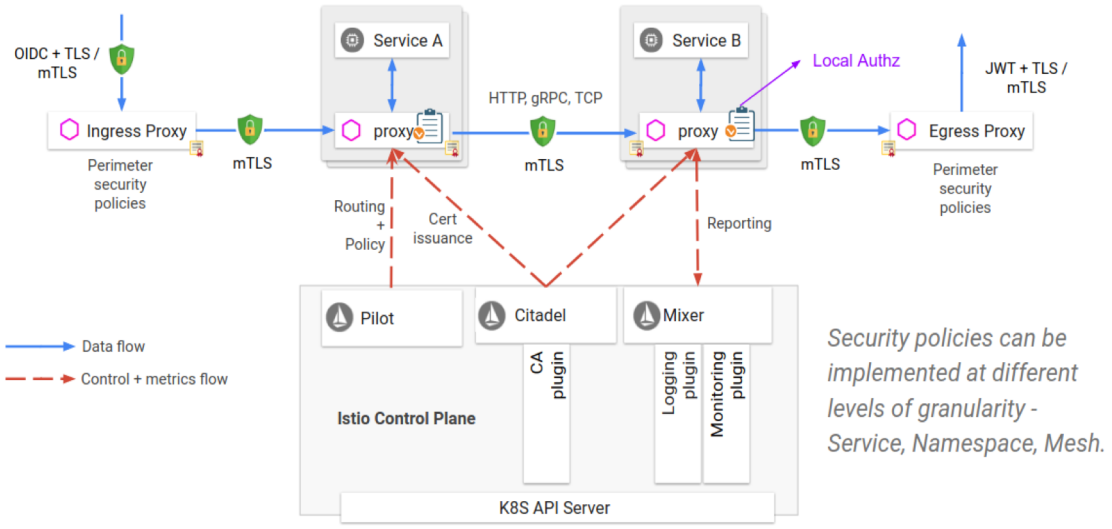
Istio Network Security

- Istio can be used to manage microservice identity, authentication, and authorization
- Identity
  - Service Account – Kubernetes resource that provides an identity for processes that run in a pod
  - Certificates generated by Istio's Citadel use namespace and service account to identify each microservice
- Authentication
  - Istio can specify authentication policies that configure services to require their callers to authenticate
  - Istio callers can authenticate using mutual authentication TLS (mTLS) encryption
  - A service can authenticate a request's origin using JSON Web Tokens (JWTs)
- Authorization
  - RBAC security enables a service to authorize which callers are allowed to access it
  - Service Role – How a mesh defines each of its RBAC roles
- Notes
  - mTLS and RBAC are independent
  - RBAC without mTLS can only identify the caller's namespace and IP address, or JWT, not its service account
  - Istio can also provide TLS termination for communication to and from the mesh

Microservices Network Security 24 © Copyright IBM Corporation 2018

IBM Training

Istio Security Architecture



IBM Training

Service Account

- Service Account – Kubernetes resource that provides an identity for processes that run in a pod
  - Pod definition optionally specifies a service account name
  - If not, the pod is automatically assigned the default service account in the same namespace
  - All replicas of the same pod template have the same service account name
- Processes in containers inside pods can contact the Kubernetes API Server
  - They are authenticated as a particular service account

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: bookinfouser
```

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: productpage-v1
spec:
  replicas: 1
  template:
    spec:
      serviceAccountName: bookinfouser
      containers:
      - name: productpage
        image: istio/bookinfo-productpage-v1:1.8.0
        . . .
```


IBM Training

SPIFFE workload identification

- SPIFFE ID URI format: `spiffe://trust-domain/path`

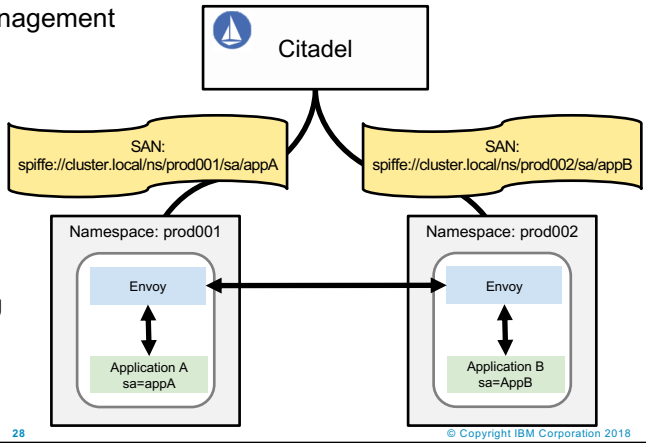
- SPIFFE – Secure Production Identity Framework For Everyone
  - Provides a secure identity, in the form of a specially crafted X.509 certificate, to every workload
  - SPIFFE IDs – Defines how services identify themselves to each other
  - SPIFFE VID – Encodes SPIFFE IDs in a cryptographically-verifiable document
  - SPIFFE Workload API – API specification for issuing and/or retrieving SVIDs
- SPIFFE Verifiable Identity Document (SVID)
  - Document that identifies a workload, can be passed to other workloads
  - Implemented as an X.509 certificate
  - SPIFFE ID is encoded in the X.509 certificate's Subject Alternative Name (SAN) extension
- SPIFFE is a Cloud Native Computing Foundation (CNCF) project
  - SPIFFE is a standard for workload identification
- SPIRE (the SPIFFE Runtime Environment) is another CNCF project, part of the SPIFFE project
  - Software that exposes the SPIFFE Workload API so workloads can retrieve their identity

Microservices Network Security 27

IBM Training

Istio Citadel (fka Istio Auth)

- Istio ID URI format (for Kubernetes)
  - `spiffe://cluster.local/ns/<namespace>/sa/<serviceaccountname>`
  - All pods with the same service account name get SPIFFE VIDs with the same identity
- Istio proxy compares SPIFFE IDs to `source.principal` and `destination.principal` fields
- Citadel is responsible for key and certificate management
  - Generation
  - Deployment
  - Rotation
  - Revocation
- Citadel certificates are SPIFFE VIDs
- Citadel is an alternative to SPIRE
  - Citadel is a more comprehensive security solution
  - Includes authentication, authorization, and auditing



Microservices Network Security 28

IBM Training

Istio authentication policies

- Use authentication policies to force services to require authentication and configure it
  - Policy can be scoped to the mesh, a namespace, or selected services
- Mesh Policy – Mesh-wide
  - Scope is all pods in the mesh
  - At most one per mesh, always named `default`, no targets
- Policy – Namespace-wide
  - Scope is all pods in a namespace
  - At most one per namespace, always named `default`, no targets
- Policy – Service-specific
  - Scope is specified pods in a namespace
  - Zero, one, or many per namespace; includes target selectors
- For each service, Istio applies the narrowest matching policy
  - service-specific > namespace-wide > mesh-wide
- Only transport authentication supported is mTLS (Istio v0.7)
- Only origin authentication supported is JWT (Istio v0.7)

```
apiVersion: authentication.istio.io/v1alpha1
kind: Policy
metadata:
  name: mTLS_enable
  namespace: prod
spec:
  target:
  - name: productpage
    ports:
    - number: 9000
  peers:
  - mtls: {}
  origins:
  - jwt:
      issuer: "https://securetoken.google.com"
      audiences:
      - "productpage"
      jwksUri: "https://.../oauth2/v1/certs"
      jwt_headers:
      - "x-goog-iap-jwt-assertion"
  principalBinding: USE_ORIGIN
```

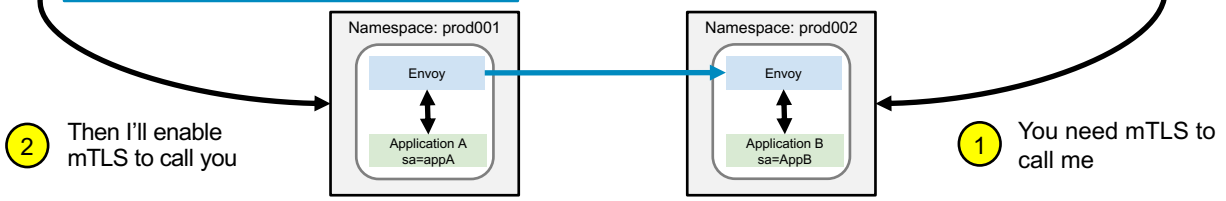

IBM Training

Enable mutual TLS authentication

- A (mesh) policy that specifies mTLS forces its services to require transport authorization via mTLS
  - Enabling mTLS policy causes Citadel to generate certificates for the Istio proxies
- Callers to these services must include a destination rule that enables mTLS

```
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: applicationB-dr
  namespace: prod002
spec:
  host: applicationB
  trafficPolicy:
    tls:
      mode: ISTIO_MUTUAL
```

```
apiVersion: authentication.istio.io/v1alpha1
kind: Policy
metadata:
  name: prod002-policy
  namespace: prod002
spec:
  peers:
  - mtls: {}
```



IBM Training

Enabling authorization

- Istio configures authorization using Role-based Access Control (RBAC)
- RBAC Config
  - Mesh-wide singleton that enables authorization, always named default
- Mode
  - OFF – Istio authorization is disabled
  - ON – Istio authorization is enabled for all services in the mesh
  - ON_WITH_INCLUSION – Istio authorization is enabled only for specified services and namespaces
  - ON_WITH_EXCLUSTION – Istio authorization is enabled for all services in the mesh except the specified services and namespaces

```
apiVersion: "rbac.istio.io/v1alpha1"
kind: RbacConfig
metadata:
  name: default
  namespace: istio-system
spec:
  mode: ON_WITH_EXCLUSION
  exclusion:
    namespaces: [ "test" ]
```

```
apiVersion: "rbac.istio.io/v1alpha1"
kind: RbacConfig
metadata:
  name: default
  namespace: istio-system
spec:
  mode: ON_WITH_INCLUSION
  inclusion:
    namespaces: [ "default" ]
```

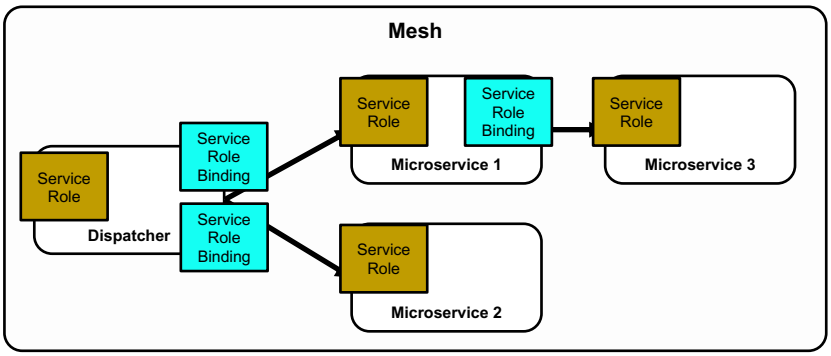

IBM Training

Authorization policy

- Service Role
  - Represents permission to invoke these services using all or parts of their API
  - Specifies a set of services, and the methods and paths for invoking those services, and other constraints
  - Services are selected by name (including wildcards) and other constraints (typically pod labels)
- Service Role Binding
  - Grants permission to invoke the services using the specified parts of their API
  - Associates one or more subjects with a service role
  - Subjects are specified by service accounts and other identity properties
- The clients with a service role binding are allowed to call the services with a service role
  - The service blocks the call if the caller doesn't have the proper binding
  - The service blocks the call if the caller tries to use parts of the API that aren't enabled
  - The service blocks the call if the request doesn't meet the specified constraints

IBM Training

Role-based authorization for microservices application

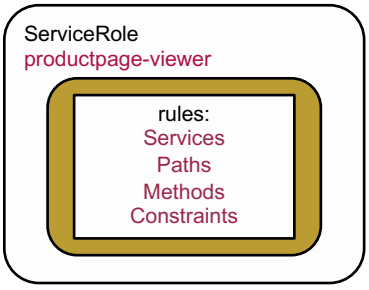


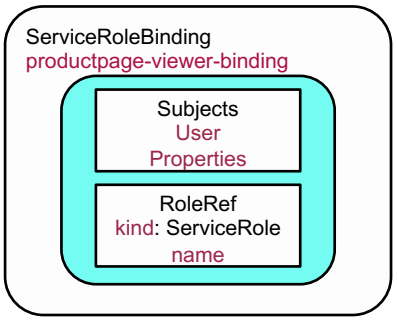
- RBAC has been enabled for this mesh and these services
- Each microservice has a service role
- Each microservice client has a service role binding for each service it's allowed to invoke

IBM Training

Role Based Access Control (RBAC) configuration

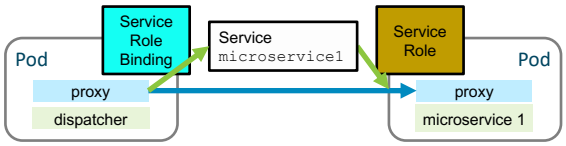
- ServiceRole
  - rules – The set of permissions
  - services – The list of services with this role
  - methods – A list of HTTP methods
  - paths – A list of HTTP paths or gRPC methods
  - constraints – Filters based on known keys
- ServiceRoleBinding
  - roleRef – Name of the ServiceRole
  - subjects – Clients of the service
  - user – Service account name
  - properties – Filters based on known keys





IBM Training

Configuring ServiceRole and ServiceRoleBinding




```
apiVersion: config.istio.io/v1alpha2
kind: ServiceRoleBinding
metadata:
  name: bind-microservice1-viewer
  namespace: appl
spec:
  subjects:
  - user: "*"
    properties:
      source.namespace: appl
  roleRef:
    kind: ServiceRole
    name: "microservice1-viewer"
```

```
apiVersion: config.istio.io/v1alpha2
kind: ServiceRole
metadata:
  name: microservice1-viewer
  namespace: appl
spec:
  rules:
  - services: [ "microservice1" ]
    methods: [ "GET" ]
```

Microservices Network Security 35

IBM Training

Constraint keys for service roles

Name	Description	Key Example	Values Example
destination.ip	Destination workload instance IP address, supports single IP or CIDR	destination.ip	["10.1.2.3", "10.2.0.0/16"]
destination.port	The recipient port on the server IP address, must be in the range [0, 65535]	destination.port	["80", "443"]
destination.labels	A map of key-value pairs attached to the server instance	destination.labels[version]	["v1", "v2"]
destination.name	Destination workload instance name	destination.name	["productpage*", "*-test"]
destination.namespace	Destination workload instance namespace	destination.namespace	["default"]
destination.user	The identity of the destination workload	destination.user	["bookinfo-productpage"]
request.headers	HTTP request headers, The actual header name is surrounded by brackets	request.headers[X-User]	["abc123"]

Microservices Network Security 36

IBM Training

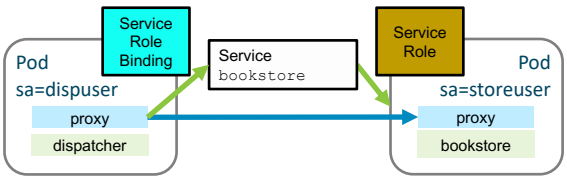
Properties keys for service role bindings

Name	Description	Key Example	Value Example
source.ip	Source workload instance IP address, supports single IP or CIDR	source.ip	"10.1.2.3"
source.namespace	Source workload instance namespace	source.namespace	"default"
source.principal	The identity of the source workload	source.principal	"cluster.local/ns/default/sa/productpage"
request.headers	HTTP request headers. The actual header name is surrounded by brackets.	request.headers[User-Agent]	"Mozilla/*"
request.auth.principal	The authenticated principal of the request	request.auth.principal	"accounts.my-svc.com/104958560606"
request.auth.audiences	The intended audience(s) for this authentication information	request.auth.audiences	"my-svc.com"
request.auth.presenter	The authorized presenter of the credential	request.auth.presenter	"123456789012.my-svc.com"
request.auth.claims	Claims from the origin JWT. The actual claim name is surrounded by brackets.	request.auth.claims[iss]	"*@foo.com"

IBM Training

Authorization considerations

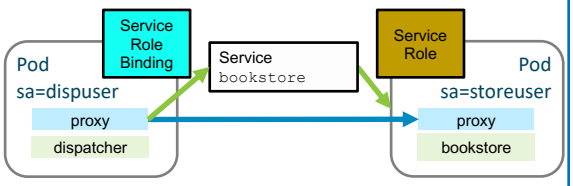
- A ServiceRole applies to Services, a ServiceRoleBinding applies to ServiceAccounts
- A single ServiceRole can be used to configure multiple services and multiple HTTP methods
- A single service can be configured with multiple ServiceRoles
  - It may be easier to configure a single service using multiple (non-conflicting) ServiceRoles
- Multiple ServiceRoles are required to configure multiple levels of access
  - Read-only: GET
  - Read/write: GET/POST
  - Administer: GET/PATCH/PUT/DELETE
- Each client needs a ServiceRoleBinding for permission to invoke a service
  - A single ServiceRoleBinding can be used to configure multiple ServiceAccounts



IBM Training

Selecting binding subjects

- Specifying service role binding subjects
  - user
  - source.principal
- Envoy interprets wildcards differently
- user
  - "*" all users including unauthenticated (no certificate)
- source.principal
  - "*" all authenticated users




```
apiVersion: rbac.istio.io/v1alpha1
kind: ServiceRole
metadata:
  name: tester
  namespace: appl
spec:
  rules:
  - services: ["test-*"]
    methods: ["*"]
  - services: ["bookstore.appl.svc.cluster.local"]
    paths: ["*/reviews"]
    methods: ["GET"]
```

```
apiVersion: rbac.istio.io/v1alpha1
kind: ServiceRoleBinding
metadata:
  name: bind-microservice1-viewer
  namespace: appl
spec:
  roleRef:
    kind: ServiceRole
    name: "tester"
  subjects:
  - user: "cluster.local/ns/app1/sa/dispuser"
    properties:
      source.principal: "cluster.local/ns/app1/sa/dispuser"
```

Microservices Network Security 39 © Copyright IBM Corporation 2018

IBM Training

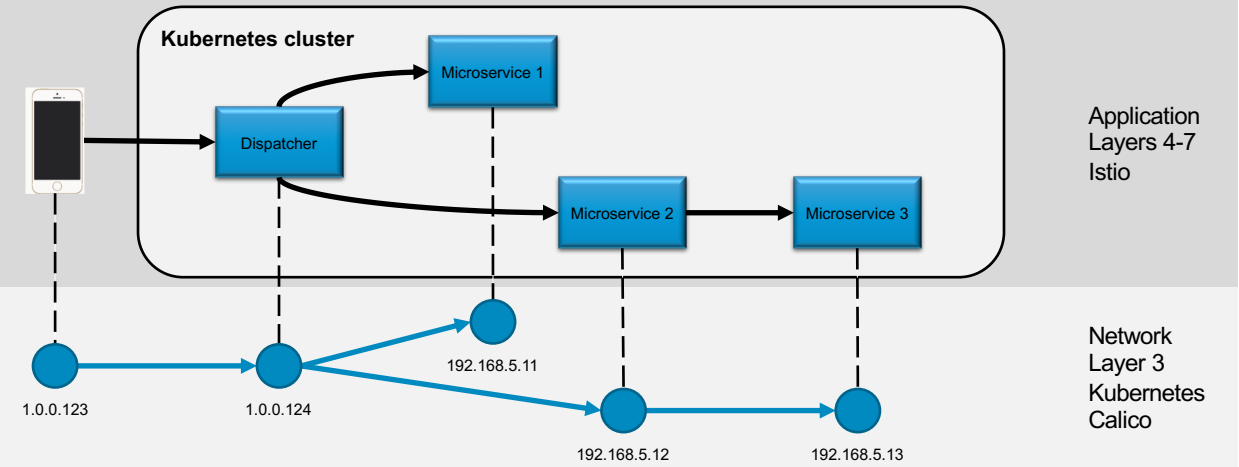
Bringing it all together

40 © Copyright IBM Corporation 2018

IBM Training

Review: Two layers of application protection

- Configure Kubernetes Network Policy (layer 3) and Istio (layers 4-7) together and consistently
- If Service A can call Service B, define that in both layers using both technologies

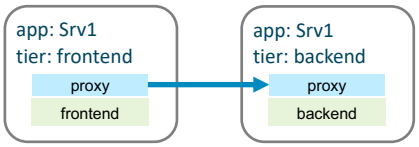


Microservices Network Security 41 © Copyright IBM Corporation 2018

IBM Training

Application protection with network policy and Istio

- Service A can call Service B
  - Enable in the network (layer 3) using Kubernetes Network Policy
  - Enable in the application (layers 4-7) using Istio authorization




```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: backend-allow
spec:
  podSelector:
    matchLabels:
      app: Srv1
      tier: backend
  ingress:
  - from:
      - podSelector:
          matchLabels:
            app: Srv1
            tier: frontend
```

```yaml
apiVersion: rbac.istio.io/v1alpha1
kind: ServiceRole
metadata:
  name: backend-allow
spec:
  rules:
  - services: [backend]
    methods: [GET, POST]
---
apiVersion: rbac.istio.io/v1alpha1
kind: ServiceRoleBinding
metadata:
  name: backend-allow
spec:
  subjects:
  - user: "cluster.local/ns/default/sa/frontend"
  roleRef:
    kind: ServiceRole
    name: backend-allow
```

Microservices Network Security 42 © Copyright IBM Corporation 2018

Conclusion

- Understand network security options
- Understand Kubernetes network policy basics
- Understand Network security with Istio

IBM Training