

Quality Attributes

- *GitLab*

Tobias, Nicolai & Torben

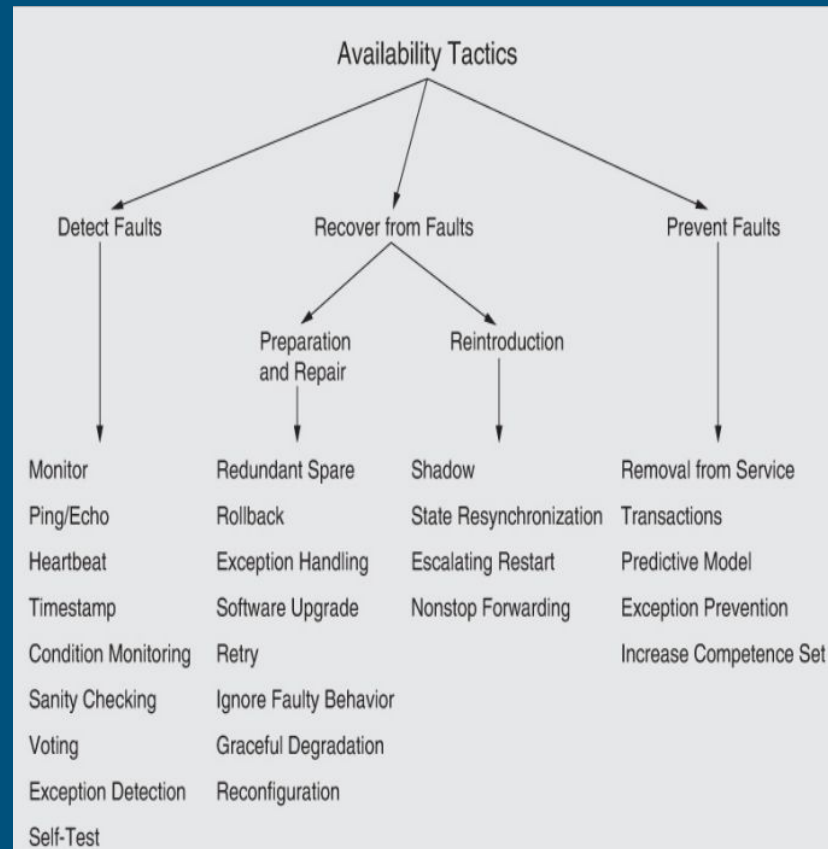
Agenda

- Availability
Tactics, Examples, Pros & Cons
 - Deployability
Tactics, Examples, Pros & Cons
 - Interoperability
Tactics, Examples, Pros & Cons
 - Quiz
Blooket
-

Availability - Tactics

Types:

- Detect Faults
- Recover from Faults
 - Preparation & Repair
 - Reintroduction
- Prevent Faults



Availability - Examples

Detect Faults

- Monitor
- Heartbeat

GET `http://localhost/-/health`

GET `http://localhost/-/readiness`

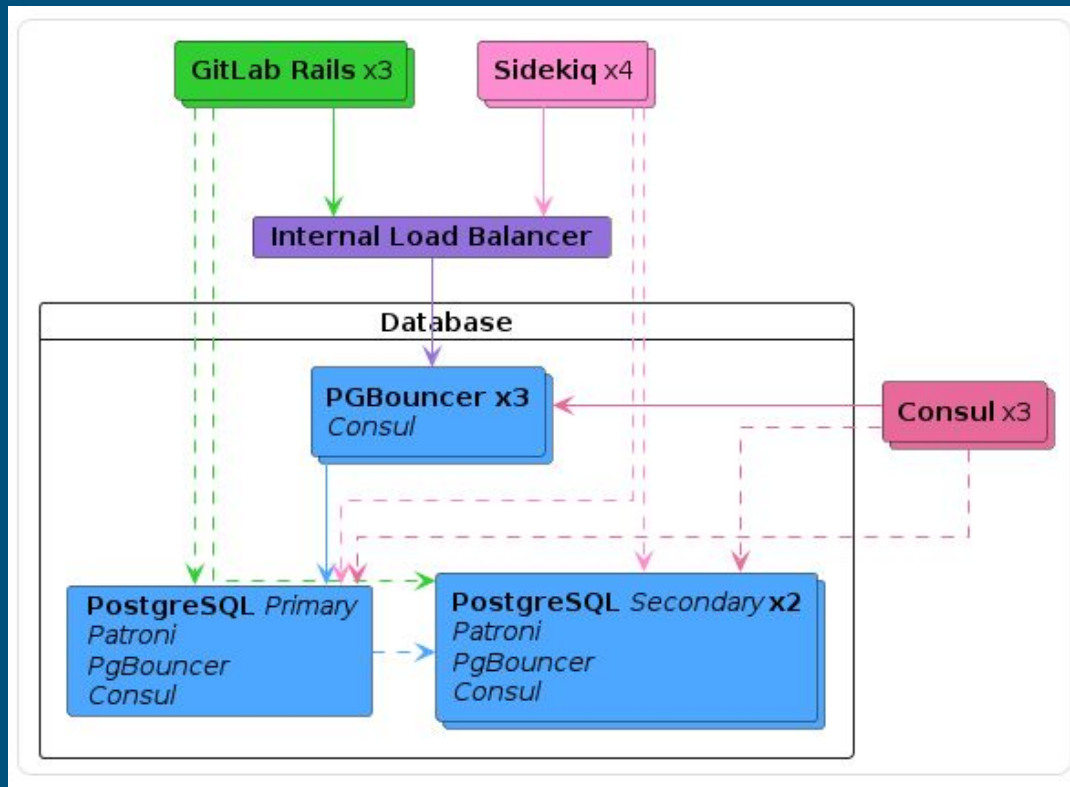
GET `http://localhost/-/liveness`

master ▾ gitlab / lib / gitlab / error_tracking			History	Find file	📄 ▾	Clone ▾
Name	Last commit	Last update				
..						
📁 error_repository	Remove remaining redundant freezes	1 month ago				
📁 processor	Update grpc to v1.55.0 and add monkey patch	3 months ago				
🔥 context_payload_generator.rb	Resolves rubocop offense Style/RescueStandardError	2 years ago				
🔥 detailed_error.rb	Improve integrated error tracking issue details	2 years ago				
🔥 error.rb	Rename Sentry::Client to ErrorTracking::SentryClient	2 years ago				
🔥 error_collection.rb	Rename Sentry::Client to ErrorTracking::SentryClient	2 years ago				
🔥 error_event.rb	Security fix sentry issue leaks and access level check	1 year ago				
🔥 error_repository.rb	Error Tracking: Remove unused ActiveRecordStrategy	4 months ago				
🔥 log_formatter.rb	Revert "Merge branch 'qmnguyen0711/849-sentry-upgrade-sentry-raven-..."	2 years ago				
🔥 logger.rb	Remove toggles in 'process_exception'	1 year ago				
🔥 project.rb	Rename Sentry::Client to ErrorTracking::SentryClient	2 years ago				
🔥 repo.rb	Rename Sentry::Client to ErrorTracking::SentryClient	2 years ago				
🔥 stack_trace_highlight_decorator.rb	Rename Sentry::Client to ErrorTracking::SentryClient	2 years ago				

Availability - Examples

Recover from Faults

- Redundant spares
- Shadow
- State Resynchronization



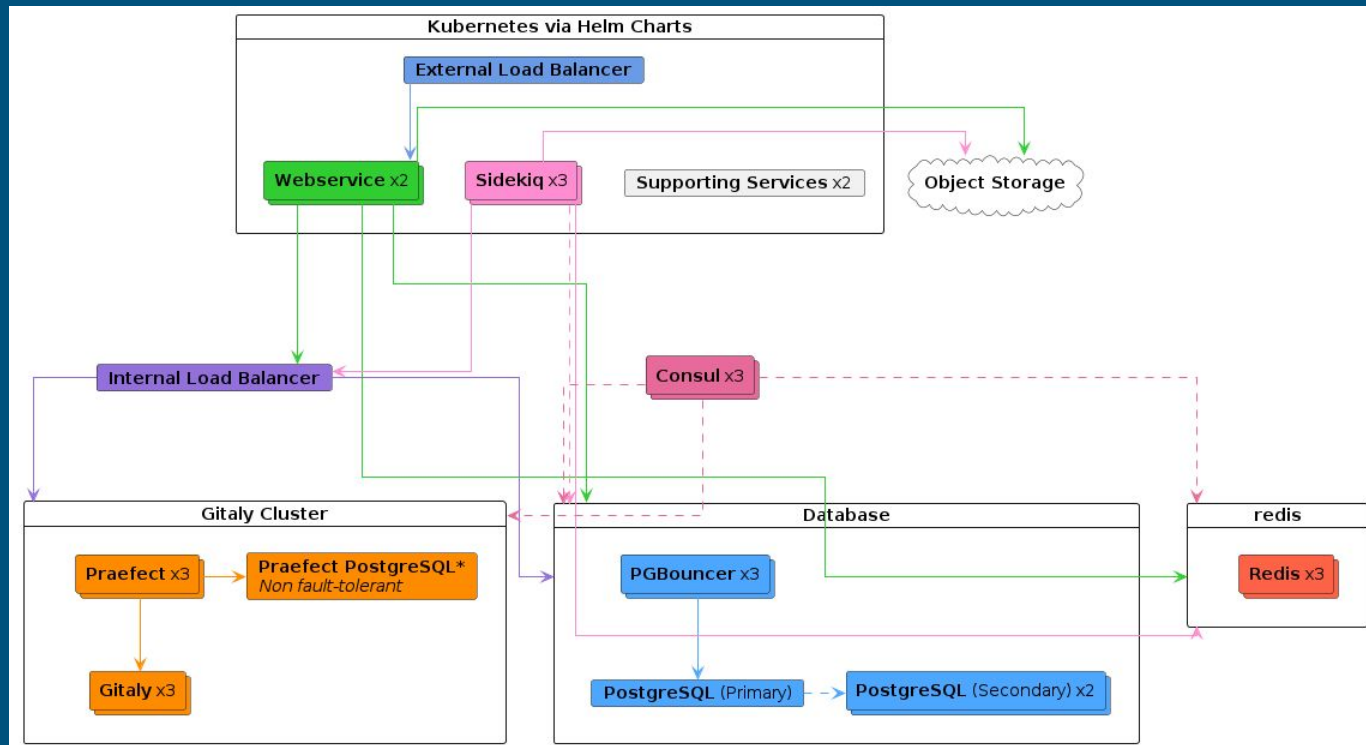
Source

Availability - Examples

Prevent Faults

- Removal from service
- Transactions

Source



Availability - Implementation Example

Redundant Spare, State Resynchronization

```
def configure_model(name)
  source_model = Gitlab::Database.database_base_models_with_gitlab_shared[name]

  @model = backup_model_for(name)

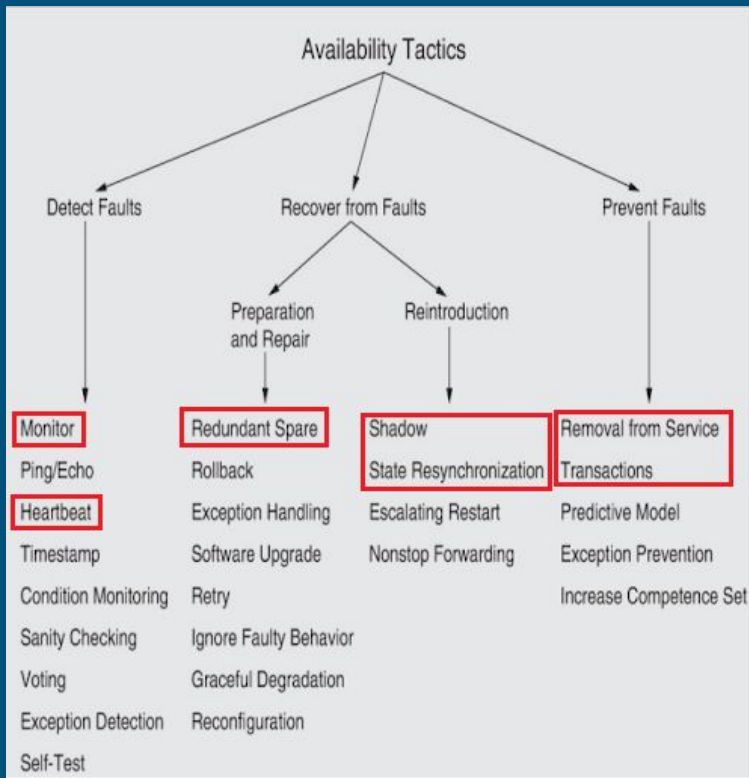
  original_config = source_model.connection_db_config.configuration_hash.dup

  @config = config_for_backup(name, original_config)

  @model.establish_connection(
    ActiveRecord::DatabaseConfigurations::HashConfig.new(
      source_model.connection_db_config.env_name,
      name.to_s,
      original_config.merge(@config[:activerecord])
    )
  )

  Gitlab::Database::LoadBalancing::Setup.new(@model).setup
end
```

Availability - Tactics



Availability - Pros & Cons

Pros

- Improved user satisfaction
 - Reliable service
 - Accessible whenever needed
 - *E-commerce website*
 - *Returning Customers*
 - *Increased sales*
- Business Continuity
 - Critical systems remain online even in the event of a failure
 - *Banking System*
 - *Maintaining trust*
 - *Financial stability*

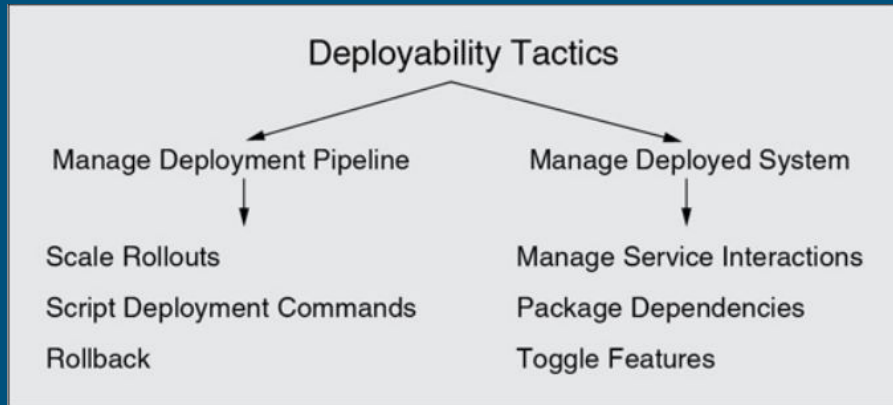
Cons

- Increased Complexity
 - Implementation of failover mechanisms
 - Introducing Redundancy
 - complex to manage and maintain
 - *Distributed Database system*
 - *Complex synchronization*
- Higher Cost
 - Redundant systems
 - Additional infrastructure
 - *Data Centers*
 - *Replicated servers geographically*
 - *Redundant power supplies*
 - *Redundant network connections*

Deployability - Tactics

Types:

- Manage Deployment Pipeline
- Manage Deployed Systems



- Script deployment commands
- Rollbacks

- sync
- preflight
- prepare
- build-images
- fixtures
- lint
- test
- post-test
- review
- qa
- post-qa
- pages
- notify
- release-environments
- benchmark

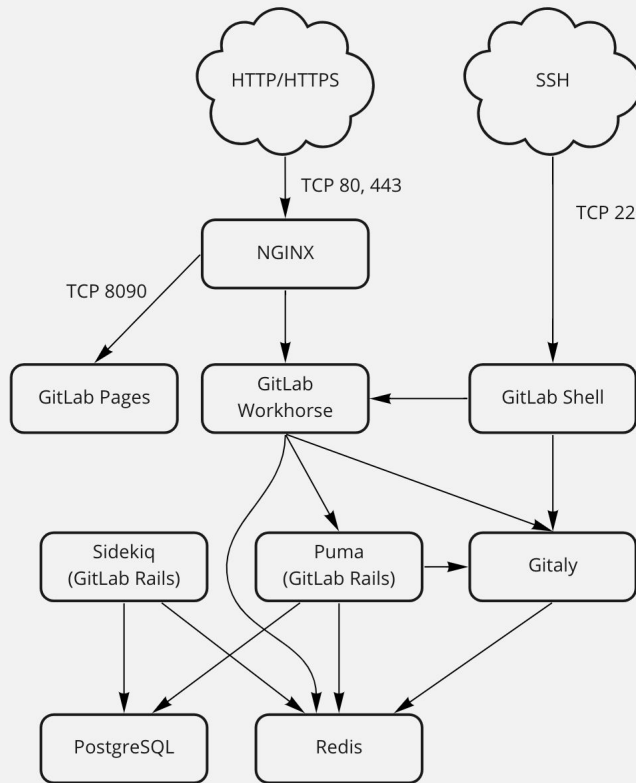
<div> <div>All 1,000+</div> <div>Finished</div> <div>Pipelines</div> <div>Tags</div> </div>			
<div> <div>Filter pipelines</div> <div>Show Pipeline ID</div> </div>			
Status	Pipeline	Created by	Stages
Created	Ruby 3.1 merged_result MR pipeline #1056469486 133902 41ce202f		<div> <div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div> </div> <div> <div></div><div></div> </div>
Running	Ruby 3.1 merged_result MR pipeline #1056464833 135581 4504be92		<div> <div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div> </div> <div> <div></div> </div>
Running	Ruby 3.1 merged_result MR pipeline #1056464273 135245 05236e9f		<div> <div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div> </div> <div> <div></div> </div>
Running	Ruby 3.1 merged_result MR pipeline #1056463158 135594 4a91aa54		<div> <div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div> </div> <div> <div></div><div></div> </div>

Deployability - Examples

Manage Deployed System

- Service interactions
 - Standalone: TCP + Unix Sockets
 - Kubernetes: K8s network
- Package dependencies
 - GitLab is written in Ruby, so most services use Gems
- Feature flags are used to toggle features

GitLab high-level architecture



All connections use Unix socket unless noted otherwise.

Deployability - Implementation Example

Script Deployment Command, Package Dependencies

cicd.yml

on: push

✓ clone-repository

25s

✓ code-analysis

40s

✓ build-application

9s

✓ test-application

10s

✓ build-push-api

34s

✓ build-push-publisher

5m 38s

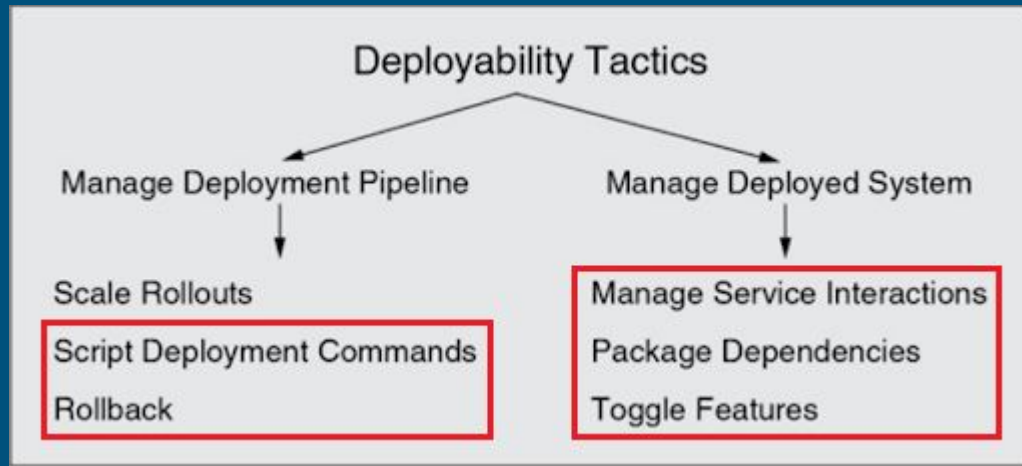
✓ build-push-subscriber

38s

✓ deploy-application

27s

Deployability - Tactics



Deployability - Pros & Cons

Pros

- Rapid Deployment and Updates
 - Quick updates and releases
 - Faster delivery of features, improvements and fixes
 - *Web application with fully automated deployment pipeline*
 - *Push updates several times a day*
 - *Minimal downtime*
 - *Ensure users have latest features*
- Consistent and Reliable Deployment Processes
 - *Reduce risk of human error*
 - *Increase consistency*
 - *Repeatable deployments (Docker)*

Cons

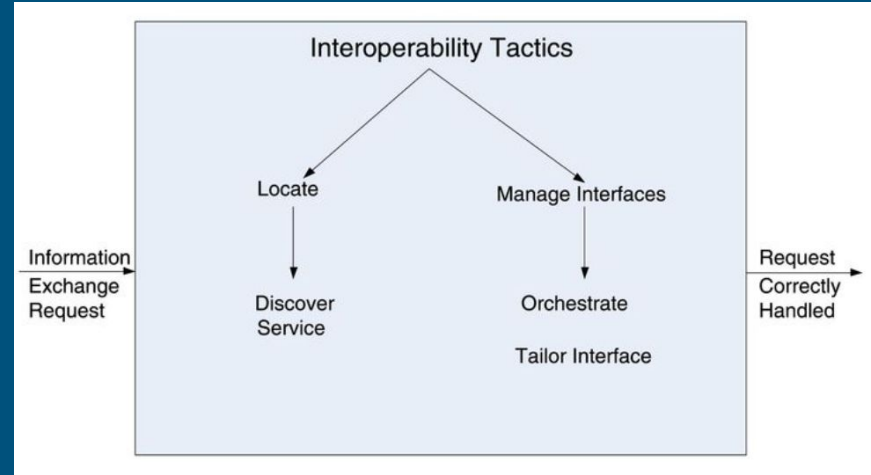
- Initial Setup Complexity and Cost
 - Automated deployment processes
 - Investment in tools, training and (potential) changes to organizational structure
 - *Continuous deployment toolchain*
 - Version control
 - Automated testing
 - Build servers
 - Deployment automation
- Ongoing Maintenance and Management
 - Ensure pipelines remain reliable
 - Tool updates and management

Interoperability - Tactics

Types:

- Locate
- Manage Interfaces

The *ability* of a system or component to *interact*, *operate* and *communicate* with other systems or components



Interoperability - Examples

Locate (*Discover Service*)

- Doesn't rely heavily on service discovery
 - *like a microservice architecture might do*
 - *Service Registry*
- Runner Discovery
 - Registration of runner during runtime
 - Self-hosted runners
 - Runners Polling
 - New work that matches their tags and configuration

Interoperability - Examples

Manage Interfaces

- CI/CD Pipelines (*Orchestrate*)
 - Jobs, Stages and rules
 - Jobs in parallel, Stages sequentially
 - *Complex interactions*
- API (*Tailor Interface*)
 - Complex endpoints tailored based on user permissions
- Webhooks (*Tailor Interface*)
 - Webhook system for tailored to trigger based on specific events
- 3rd party integrations
 - Tailor settings for integrations

Project integrations

Akismet

Apple App Store

Arkose Protect

Asana

Atlassian Bamboo

AWS CodePipeline

Datadog

Diagrams.net

Discord Notifications

Elasticsearch

Emails on push

External issue trackers

GitHub

GitLab for Slack app

Gitpod

Gmail actions

Google Chat

Google Play

Harbor

irker (IRC gateway)

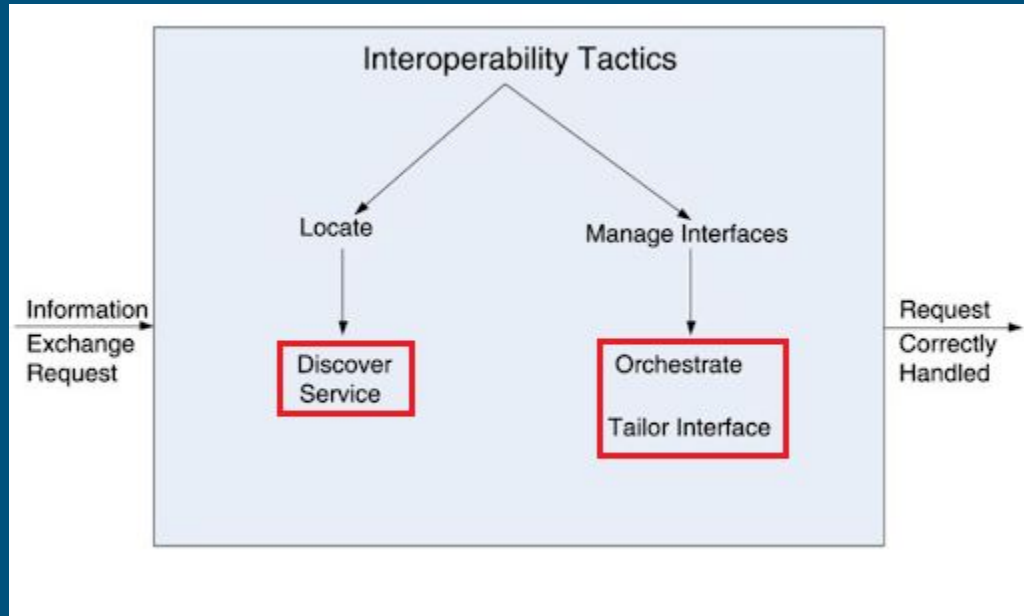
Jenkins

Interoperability - Implementation Example

Discover Service

```
1 using System.Collections.Concurrent;
2
3 public class ServiceRegistry
4 {
5     private readonly ConcurrentDictionary<string, Uri> _services = new ConcurrentDictionary<string, Uri>();
6
7     public bool RegisterService(string serviceName, Uri serviceUri)
8     {
9         return _services.TryAdd(serviceName, serviceUri);
10    }
11
12    public bool DeregisterService(string serviceName)
13    {
14        return _services.TryRemove(serviceName, out _);
15    }
16
17    public Uri DiscoverService(string serviceName)
18    {
19        _services.TryGetValue(serviceName, out var serviceUri);
20        return serviceUri;
21    }
22
23    public IEnumerable<KeyValuePair<string, Uri>> GetAllServices()
24    {
25        return _services.ToArray();
26    }
27 }
```

Interoperability - Tactics



Interoperability - Pros & Cons

Pros

- Seamless Integration
 - Integration between software systems
 - Improved functionality & dataflow across enterprise
 - *Between enterprise and external partners*
 - *Enterprise Resource Planning (ERP)*
 - *Integration with many external vendors*
- Flexibility and Scalability
 - Easier to scale and modify to meet new business needs
 - Add, upgrade and replace components
 - *Microservice Architectures*
 - *Scale individual services independently*
 - *Wordpress (CMS)*
 - *Interoperable plugins and themes*

Cons

- Complexity in design and testing
 - Designed to comply with common standards
 - Sophisticated interfaces and extensive testing
 - *Adhering to multiple industry standards*
 - *Government application*
- Potential for security Vulnerabilities
 - Open communication protocols (Exposure)
 - *Sharing data across organization boundaries*
 - *More susceptible to data breaches*
 - *Each integration point is a potential attack vector*

~~Kahoot!~~

Blooket