Quality Attributes

- *GitLab* 

Tobias, Nicolai & Torben

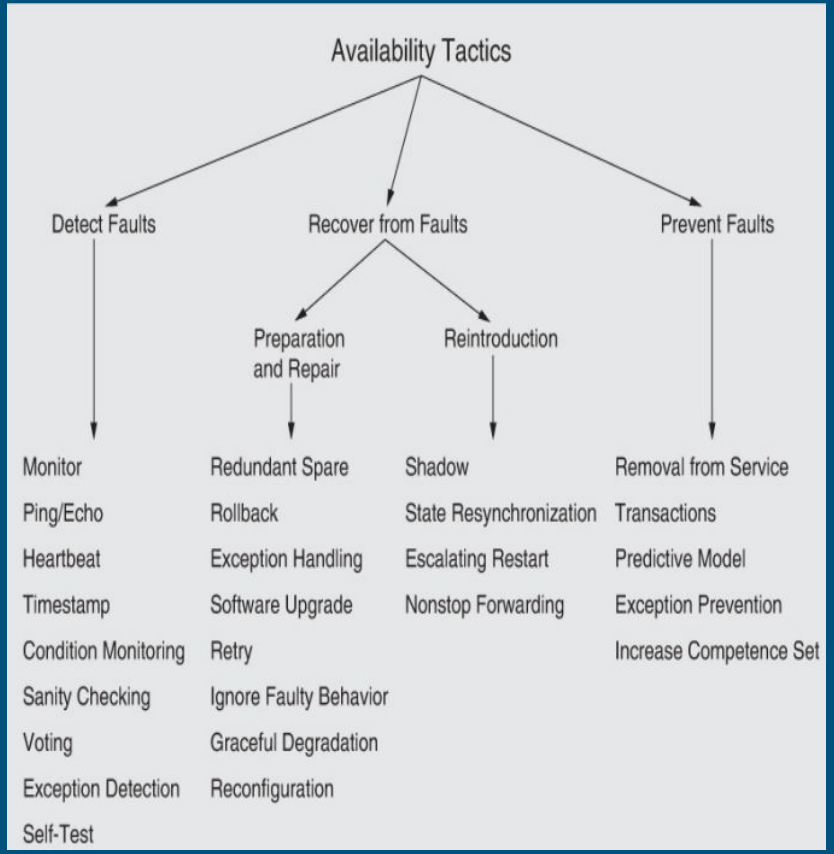
Agenda

- 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

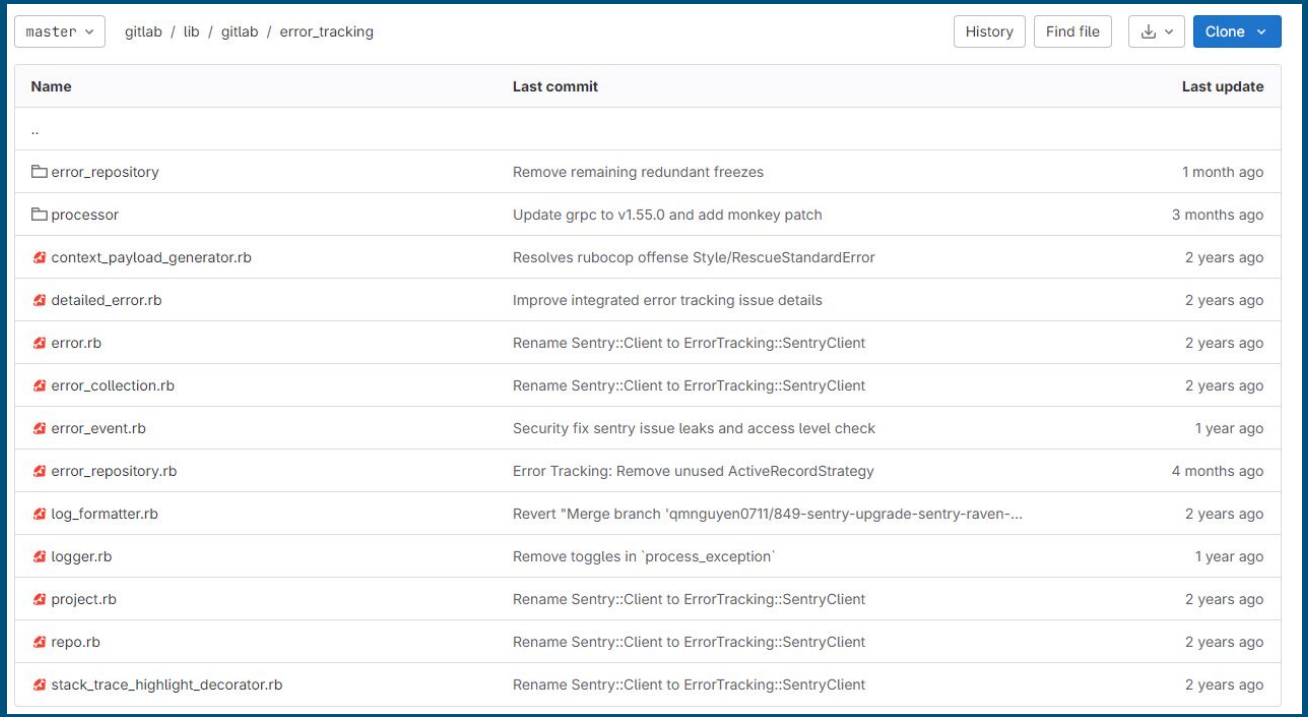
Detect Faults

- Monitor
- Heartbeat

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

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

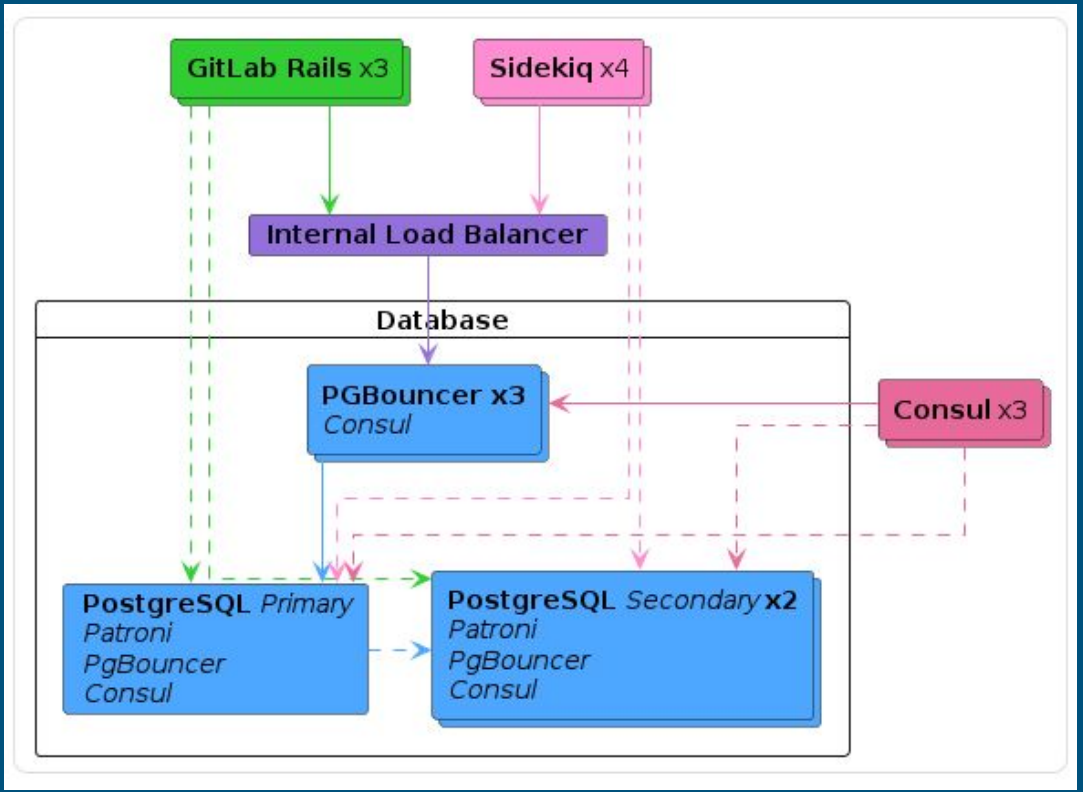
```
GET http://localhost/-/liveness
```



Availability - Examples

Recover from Faults

- Redundant spares
- Shadow
- State Resynchronization

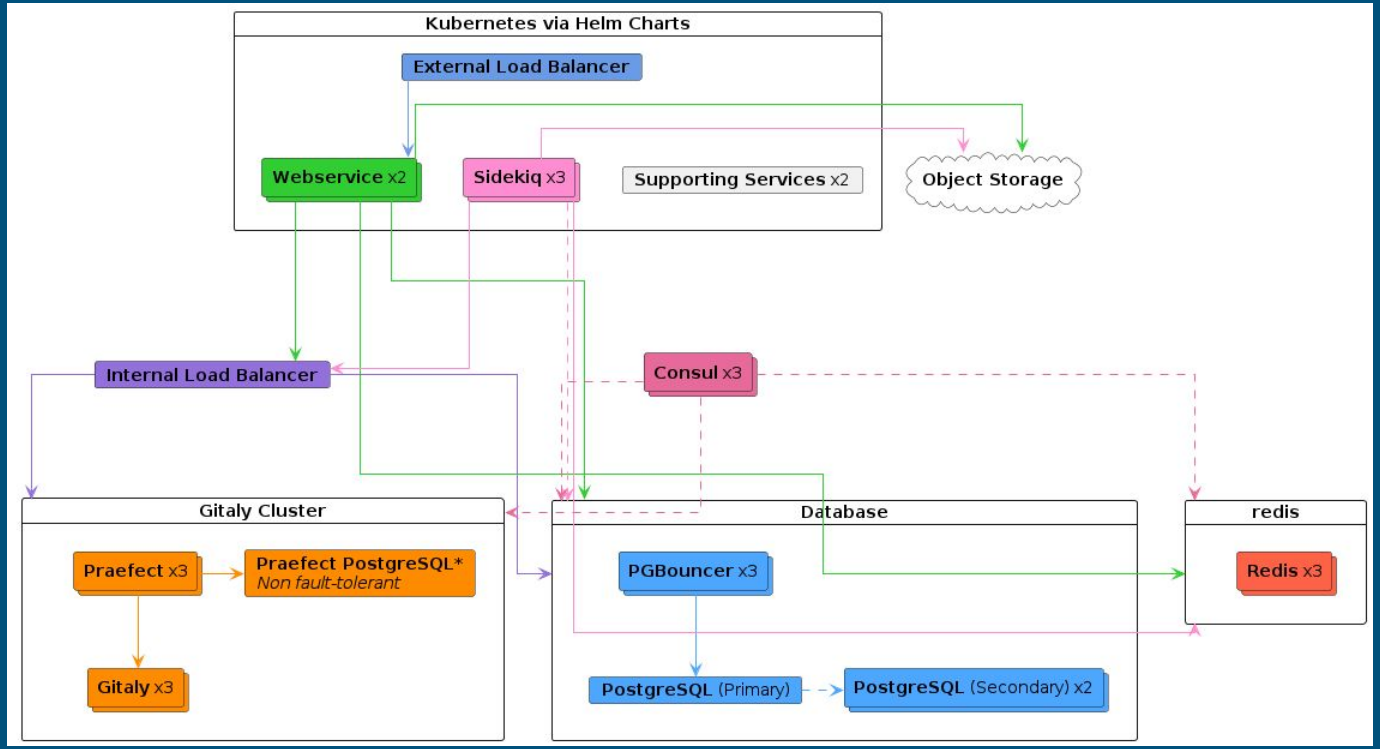


Source

Availability - Examples

Prevent Faults

- Removal from service
- Transactions



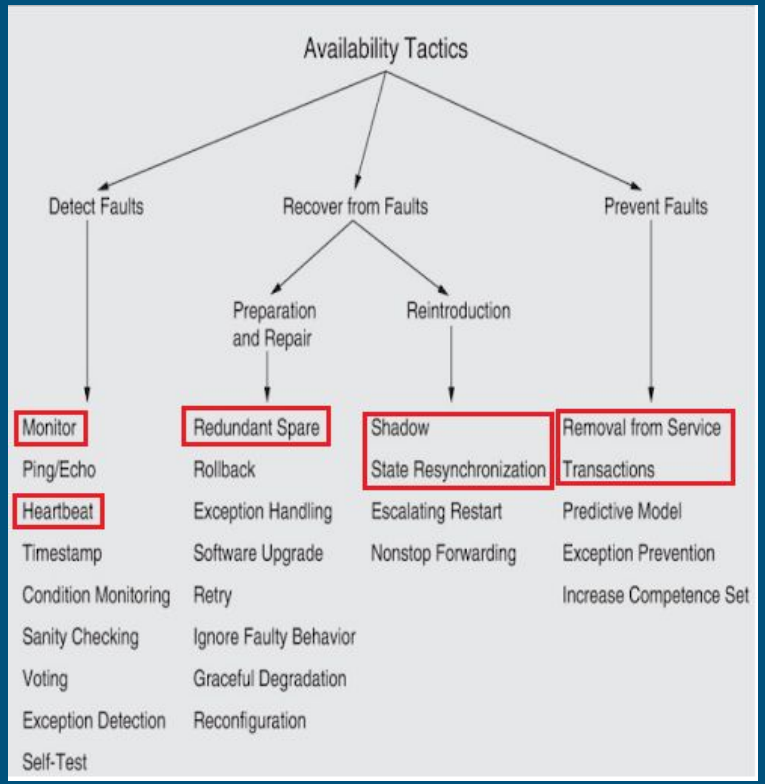
Source

Availability - Implementation Example

Redundant Spare, State Resynchronization

```ruby
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*

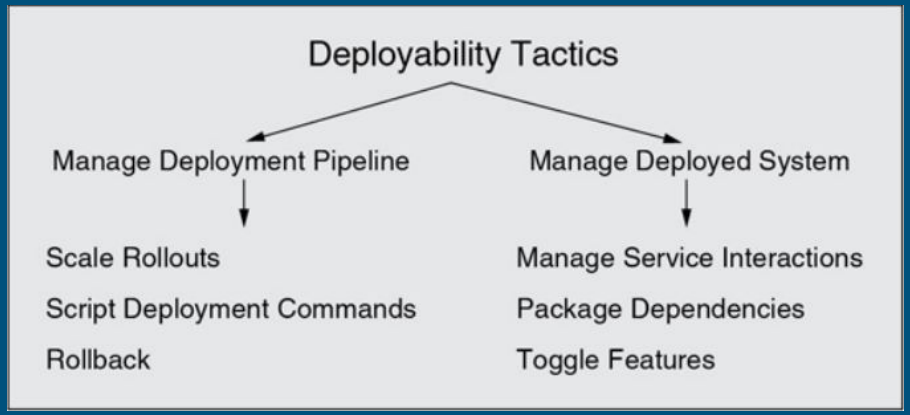
Cons

- 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

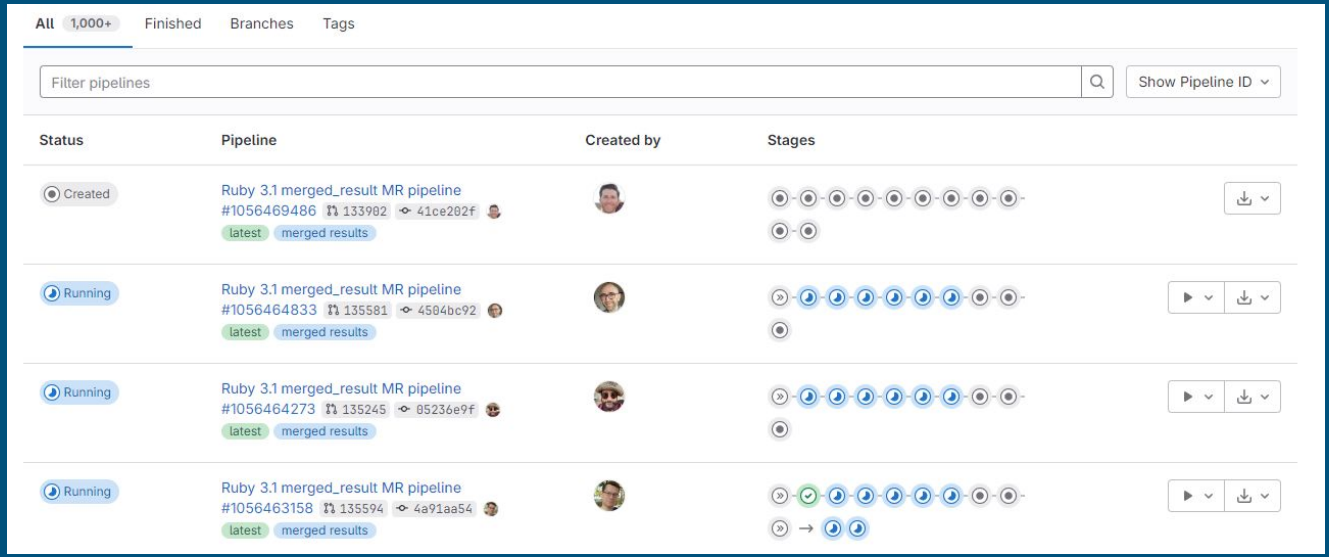


Deployability - Examples

Manage Deployment Pipeline

- Script deployment commands
- Rollbacks

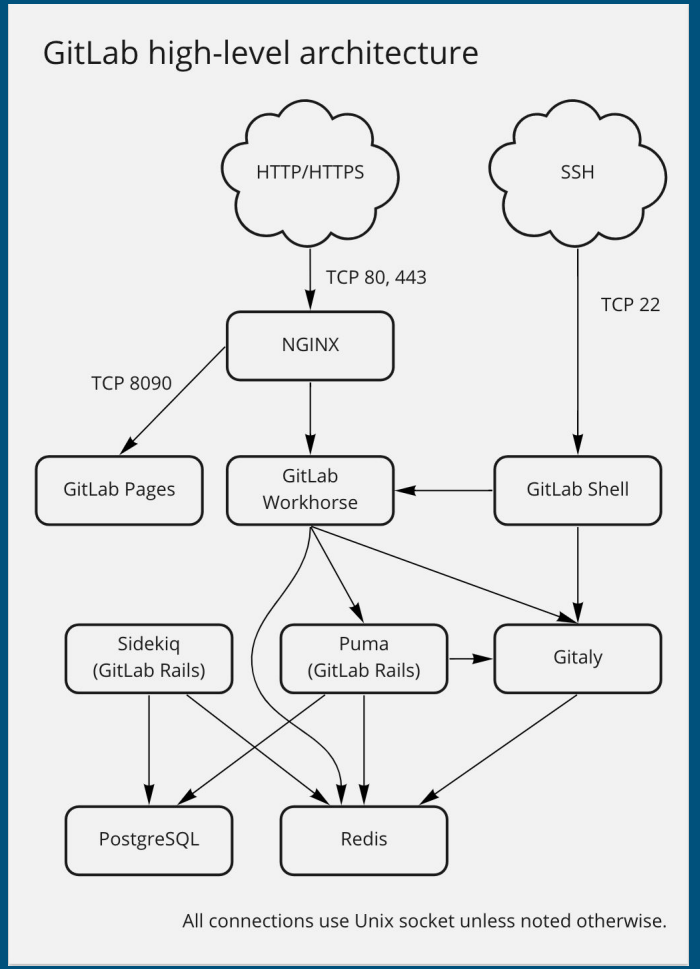
```
stages:
  - sync
  - preflight
  - prepare
  - build-images
  - fixtures
  - lint
  - test
  - post-test
  - review
  - qa
  - post-qa
  - pages
  - notify
  - release-environments
  - benchmark
```



Deployability - Examples

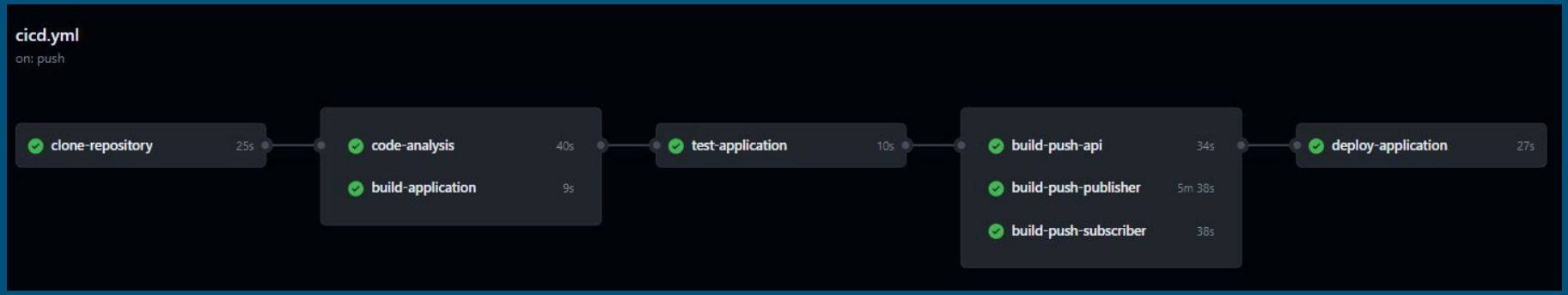
Manage Deployed System

- 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

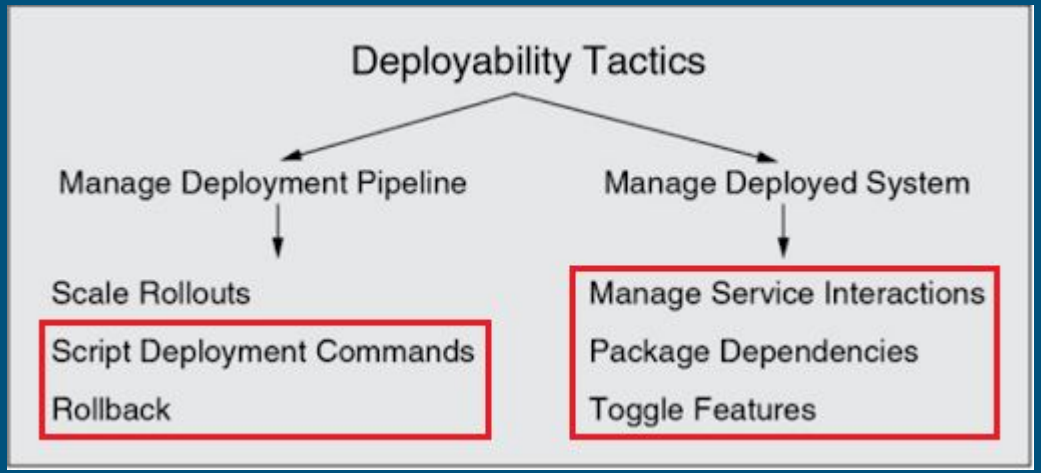


Deployability - Implementation Example

Script Deployment Command, Package Dependencies



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

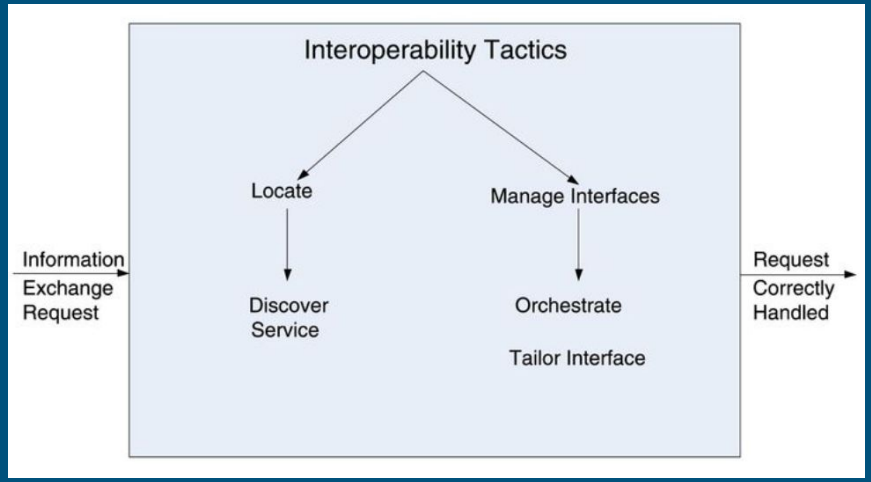
- Initial Setup Complexity and Cost
  - Automated deployment processes
  - Investment in tools, training and (potential) changes to organizational structure
  - *Continuous deployment toolchain*
    - Version control
    - Automated testing
    - Build servers
    - Deployment automation
- Ongoing Maintenance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

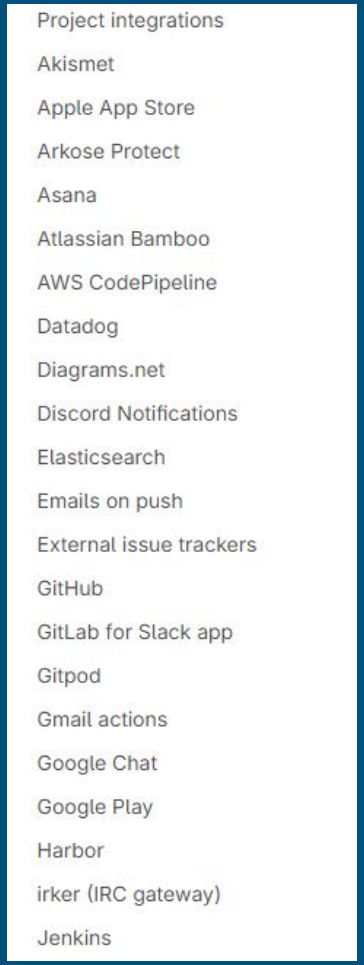
Locate (*Discover Service*)

- 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

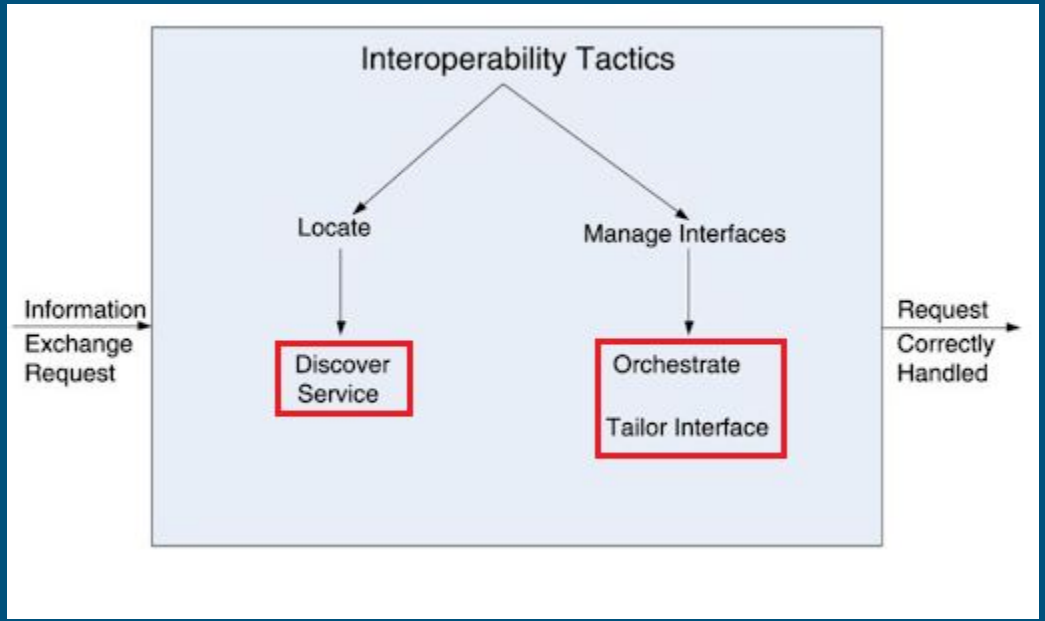


Interoperability - Implementation Example

Discover Service

```
using System.Collections.Concurrent;

public class ServiceRegistry
{
  private readonly ConcurrentDictionary<string, Uri> _services = new ConcurrentDictionary<string, Uri>();

  public bool RegisterService(string serviceName, Uri serviceUri)
  {
    return _services.TryAdd(serviceName, serviceUri);
  }

  public bool DeregisterService(string serviceName)
  {
    return _services.TryRemove(serviceName, out _);
  }

  public Uri DiscoverService(string serviceName)
  {
    _services.TryGetValue(serviceName, out var serviceUri);
    return serviceUri;
  }

  public IEnumerable<KeyValuePair<string, Uri>> GetAllServices()
  {
    return _services.ToArray();
  }
}
```

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*

Blooket