

Immigration, Refugees and Citizenship Canada: Architecture Review & Recommendations

Prepared For:

Immigration, Refugees and Citizenship Canada

Nov 2023

Revision 0.92

Red Hat Consulting - Confidential / Restricted Distribution

TABLE OF CONTENTS

[Table of Contents](#)

[Document Information](#)



[Originator](#)

[Owner](#)

[Copyright](#)

[Distribution](#)

[Confidentiality](#)

[Additional Copies](#)

[Purpose](#)

[Red Hat Consulting Contact Information](#)

[Recommendations - HA, DR & Blue/Green Deployments](#)

[Recommendations - HIP Application Architecture](#)

[Recommendations - 3Scale Architecture](#)

[Recommendations - Application Builds, Image Promotion & Configuration management](#)

[High Availability & Disaster Recovery](#)

[Blue/Green Deployments](#)

[HIP Application Architecture - Kafka Settings](#)

[3Scale Architecture](#)

[Conceptual Overview](#)

[Multi Tenancy](#)

[3Scale Deployment Architecture](#)

[3Scale Production Recommendations](#)

DOCUMENT INFORMATION

Originator

Red Hat Consulting

Owner

Red Hat Consulting - Confidential / Restricted Distribution

Copyright

This document contains proprietary information which is for exclusive use of Red Hat, Inc. and is not to be shared with personnel other than Red Hat, Inc. This document, and any portion thereof, may not be copied, reproduced, photocopied, stored electronically on a retrieval system, or transmitted without the express written consent of the owner.

Red Hat Consulting does not warrant this document to be free of errors or omissions. Red Hat Consulting reserves the right to make corrections, updates, revisions, or changes to the information contained herein. Red Hat Consulting does not warrant the material described herein to be free of patent infringement.

Unless provided otherwise in writing BY RED HAT Consulting, the information and programs described herein are provided "as is" without warranty of any kind, including but not limited to the implied warranties of merchantability and fitness for a particular purpose. In no event will RED HAT Consulting, its officers, directors, or employees or affiliates of RED HAT Consulting, their respective officers, directors, or employees be liable to any entity for any special, collateral, incidental, or consequential damages, including without any limitation, for any lost profits or lost savings, related or arising in any way from or out of the use or inability to use the information or programs set forth herein, even if it has been notified of the possibility of such damage by the purchaser or any third party.

Distribution

Do not forward or copy without written permission from Red Hat Consulting.

Copies of this document are restricted to the following names:

Red Hat, Inc.

IRCC

Confidentiality

All information supplied to Innovapost Inc Solutions for the purpose of this engagement is to be considered Red Hat confidential.

Additional Copies

Additional copies of this document can be obtained from the Service Delivery Manager listed in the [Red Hat Consulting Contact Information](#) section.

PURPOSE



This documents captures the various architecture discussions and decisions conducted during the architecture review engagement with IRCC.

RED HAT CONSULTING CONTACT INFORMATION




The table below contains the contact information for the Red Hat Consulting personnel that supported the delivery of this consulting engagement.

Role	Name	Email
Territory Services Manager	Shuzan Fawzan	sfawzan@redhat.com
Project Manager	Jasmina Jovic	jjovic@redhat.com
Architect - App Dev	Rajith Muditha Attapattu	rajith@redhat.com
Architect - Infra	Phil Thomson	pthomson@redhat.com
Sr Consultant	Casey Robb	carobb@redhat.com
Sr Consultant	Abdulahi Ali	abali@redhat.com

RECOMMENDATIONS - HA, DR & BLUE/GREEN DEPLOYMENTS


- For disaster recovery, placing the second OpenShift cluster in a different AWS data center will mitigate any outage within the primary AWS data center. 
- For A/B (Blue/Green) deployments, we recommend using ArgoRollouts when Red Hat announces General Availability. 
- Until such time, we suggest the following [label based approach](#).

RECOMMENDATIONS - HIP APPLICATION ARCHITECTURE



- The current architecture relies on Kafka to provide reliable request processing, making it a  single-point-of-failure. It is recommended to take the following mitigating steps to improve reliability.
- To improve reliability, a snapshot is written to ElasticSearch. However the microservices forwards the write request via Kafka, which creates a single-point-of-failure. Consider writing the snapshot to a database.
- A single Kafka cluster is used for both logging and business transactions. Consider using separate Kafka clusters to further improve reliability. 
- The given application flows will result in duplicates due to errors and replay mechanisms. Therefore it's strongly recommended to have duplication detection via an idempotent barrier. The current application reviewed already takes this into account. This note is more aim at future applications.
- As more applications are onboarded, it is important to segregate workloads into multiple Kafka clusters to spread the risk and mitigate a single-point-of-failure. 
- The microservices currently use XA transactions which doesn't provide any additional guarantee over local transactions since only a single resource is involved. Using local transactions will provide better performance.

- Recommended [Kafka settings](#)

RECOMMENDATIONS - 3SCALE ARCHITECTURE

- Use an external database and Redis cluster for production deployments
- Use external APICast gateways 
- Consider using an APICast gateway per group of related applications to segregate workloads

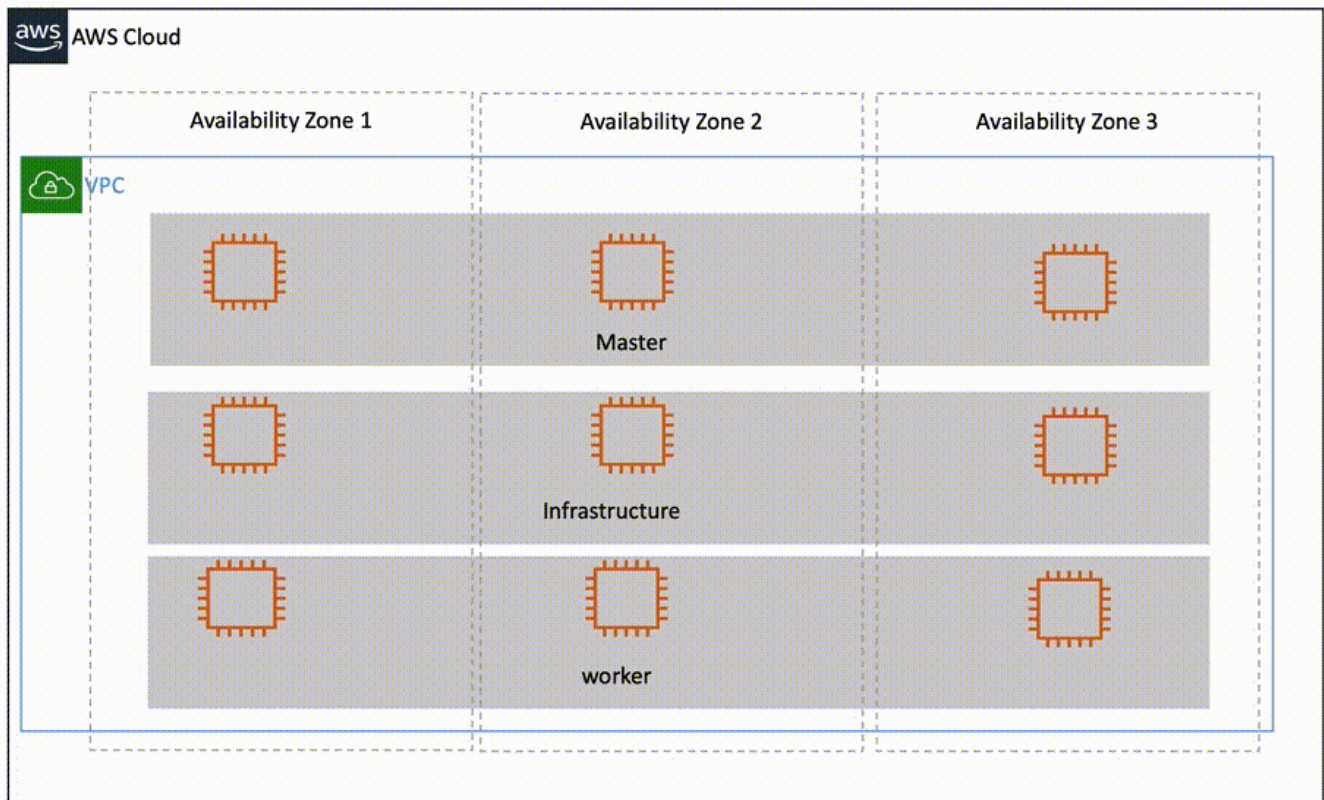
RECOMMENDATIONS - APPLICATION BUILDS, IMAGE PROMOTION & CONFIGURATION MANAGEMENT

- It was observed that secrets were stored as values in Helm charts placed in git repositories. It is understood to be a temporary measure. It is recommended to switch to using External Secrets as soon as possible. 
- While it's good practice to promote an image across environments, it is recommended to build that image in a secure and repeatable manner. Currently, this image is built in the dev environment which has wider access. It is recommended to build it in an environment with limited access. 
- It is recommended to use Tekton chains for image signing to further improve security. This technology is currently in tech preview. See here for more information.
- It is recommended to use image scanning as part of the pipeline. You could use a Tekton task that leverages Red Hat Advanced Cluster Security (ACS) for image scanning. ACS is available to you as part of OpenShift plus subscription.
- We recommend splitting infrastructure repositories to segregate higher from lower environments for

security reasons. It is currently being implemented.

HIGH AVAILABILITY & DISASTER RECOVERY

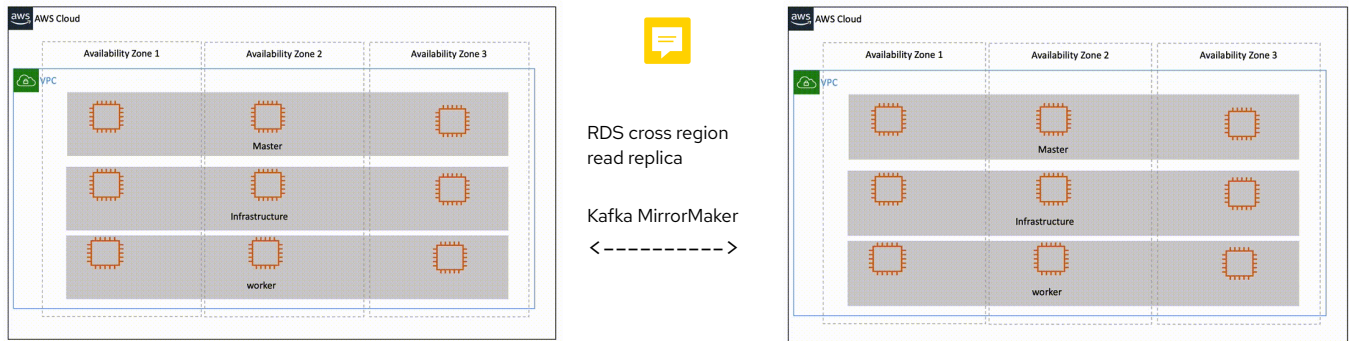
The OpenShift clusters are deployed across 3 availability zones within a data center, making it highly resilient to outages in a particular availability zone than a deployment over a single availability zone. Currently the OpenShift clusters are deployed using the below recommended reference architecture.



For improved disaster recovery, a secondary OpenShift cluster can be setup in a new data center. This allows the ability to mitigate a complete data center outage.

AWS Canada Central

AWS Canada West (early 2024)



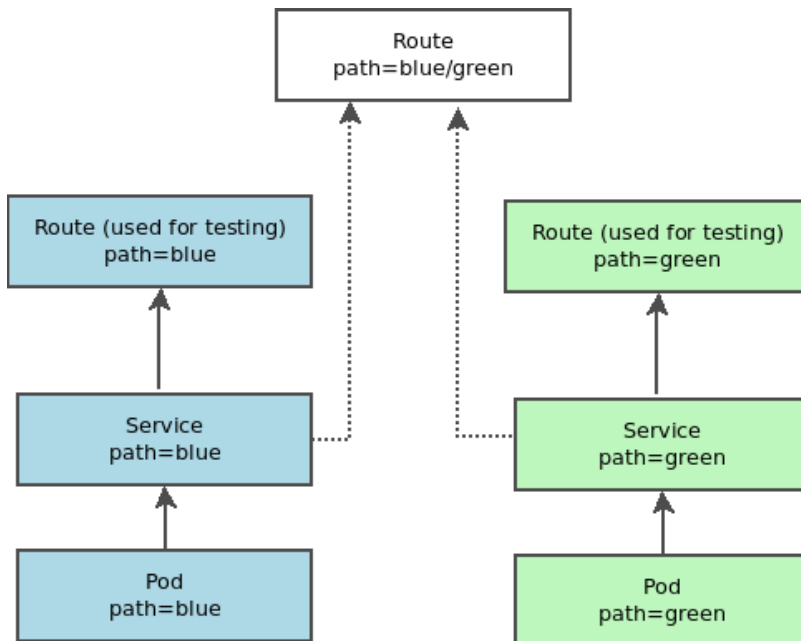
Please refer to AWS [documentation](#) to understand the reliability guarantees provided by Cross-Region read replicas.

Duplicate messages are possible with Kafka MirrorMaker due to the inherent challenges with cross site replication. Therefore it's recommended to have an idempotent barrier in the consuming applications.

BLUE/GREEN DEPLOYMENTS

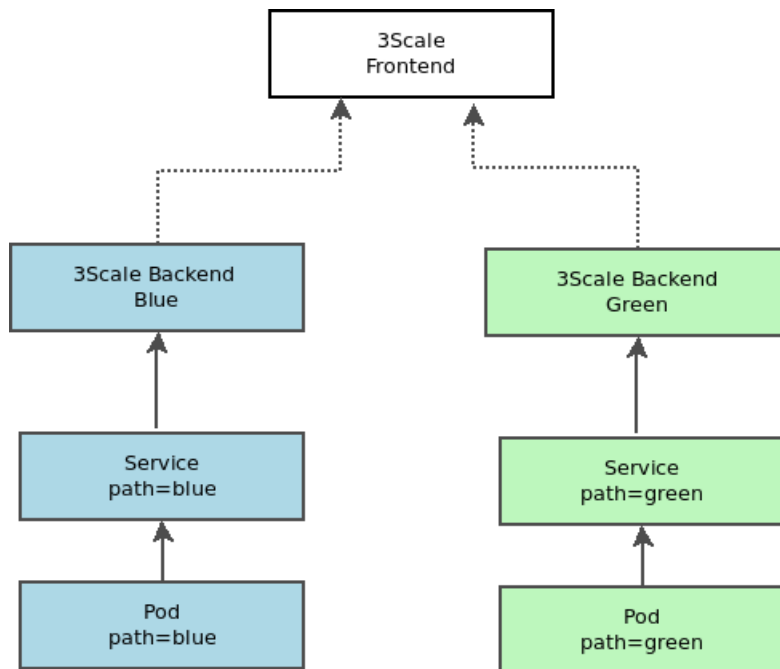
ArgoRollout is the recommended approach to handle blue/green deployments. However the Red Hat supported ArgoRollouts GA release is upcoming. Until then the following label based approach can be used.

The label based approach relies on the Kubernetes label selectors to isolate one set of services from the other as illustrated in the following diagram.



- The current version of the version of the application is running in namespace (Ex hip-xxxx-green).
- The deployments and pods in that namespace will have the label app.path=green
- A new version of the application is deployed into another namespace (Ex hip-xxxx-blue).
- The deployments and pods in that namespace will have the label app.path=blue
- Kubernetes Services uses label selectors to match pods. The Service is in the hip-xxxx-green namespace will use the label selector app.path=green to be matched to pods that have app.path=green
- Two routes will be created that points to each service.
- This allows these applications to be tested without affecting the previously deployed version.
- Once ready to go live, the global load balancer can point to the hostname that points to the ingress which points to the green path.

The same concept applies to application endpoints exposed through 3Scale as illustrated below



How to handle the common services and legacy applications?

For applications that exist outside of OpenShift (GCMS etc..), it is assumed that there exists the capability to have a parallel stack to support this approach. Our understanding is that this capability is there to support the current deployment approach.



For infrastructure services (ex Kafka, DataGrid ..etc) that are running on OpenShift the following is recommended for simplicity and to ensure the currently operational path is not affected during deployment and testing.

- Each path to have it's own Kafka cluster. This allows the configuration of topics to be the same and only the connection URLs to change. Additionally there will be no performance impact from testing.
- Each path to have it's own DataGrid cache instance. This ensures no performance impact from testing.

HIP APPLICATION ARCHITECTURE - KAFKA SETTINGS

For production the following settings are recommended. To better understand capacity it is recommended to performance test with 1.5x to 2x of the highest expected load to account for unexpected surge in traffic.

- 3 Zookeeper nodes spread across 3 availability zones using constraints and pod AntiAffinity rules
- 3 Kafka nodes spread across 3 availability zones using constraints and affinity rules
- min.insync.replicas are set to 2
- All producers to use acks=all (which is the default in the camel version used by HIP)
- Use consumer groups which preserve the offset.
- Always use the type: jbod storage for Kafka

Evenly distribute Kafka and Zookeeper nodes across multiple availability zones by using topology spread constraints in the pod spec template.

Unset

topologySpreadConstraints:

– **maxSkew:** 1

topologyKey: kubernetes.io/zone

Use podAntiAffinity to ensure no two Kafka nodes or Zookeeper nodes are not scheduled on the same node.

Unset

spec:

affinity:

podAntiAffinity:

requiredDuringSchedulingIgnoredDuringExecution:

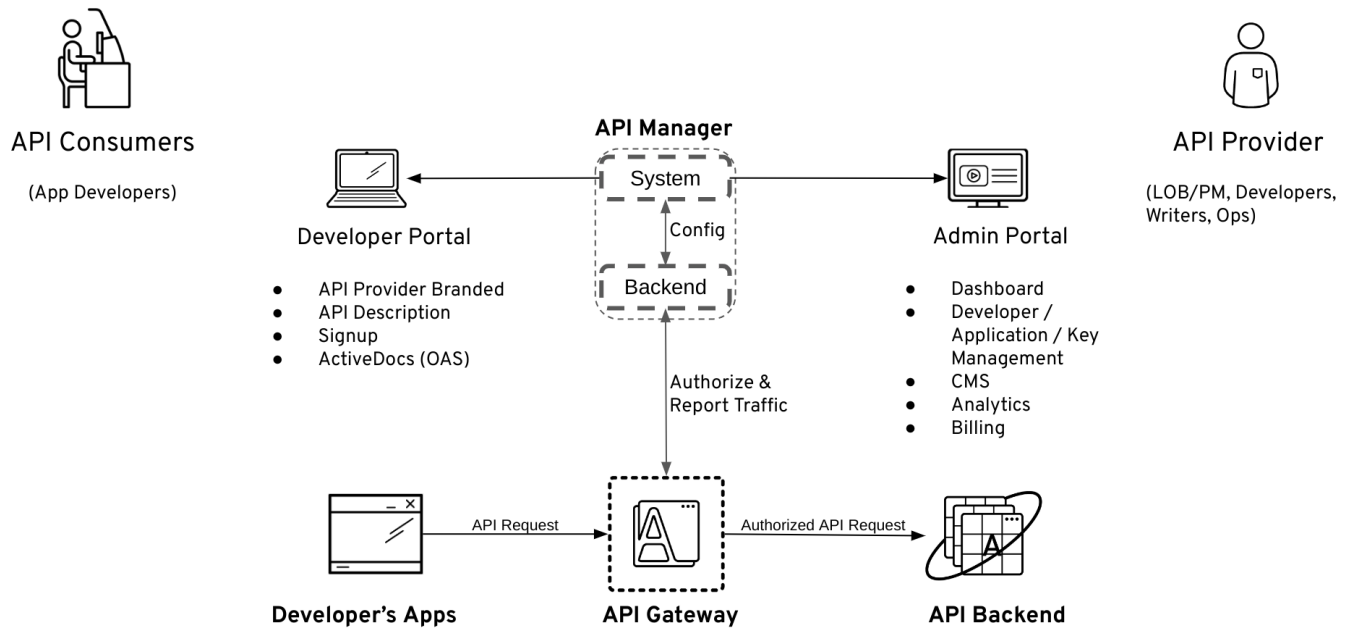
– **labelSelector:**

...

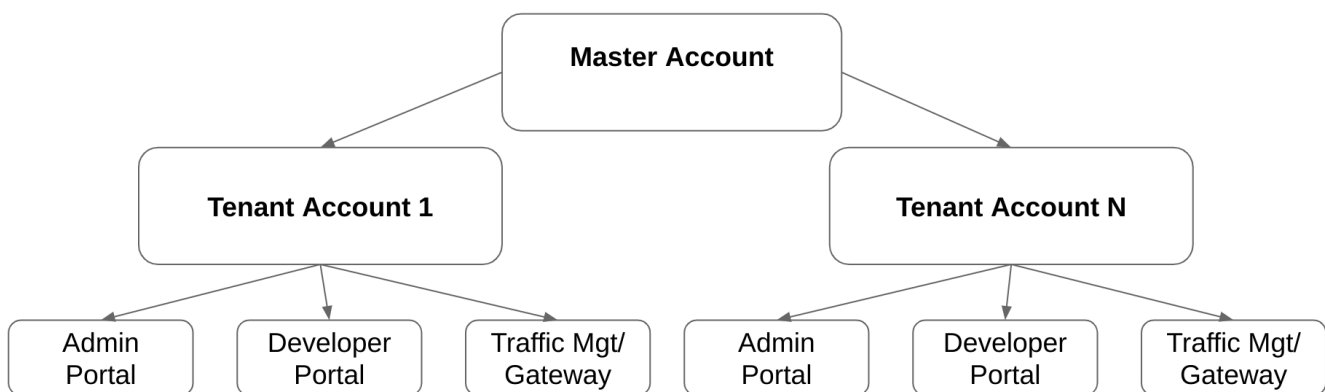
topologyKey: "kubernetes.io/hostname"

3SCALE ARCHITECTURE

Conceptual Overview



Multi Tenancy



3Scale multi tenancy provides isolation (Ex separate LoBs).

Multi Tenancy Use Cases

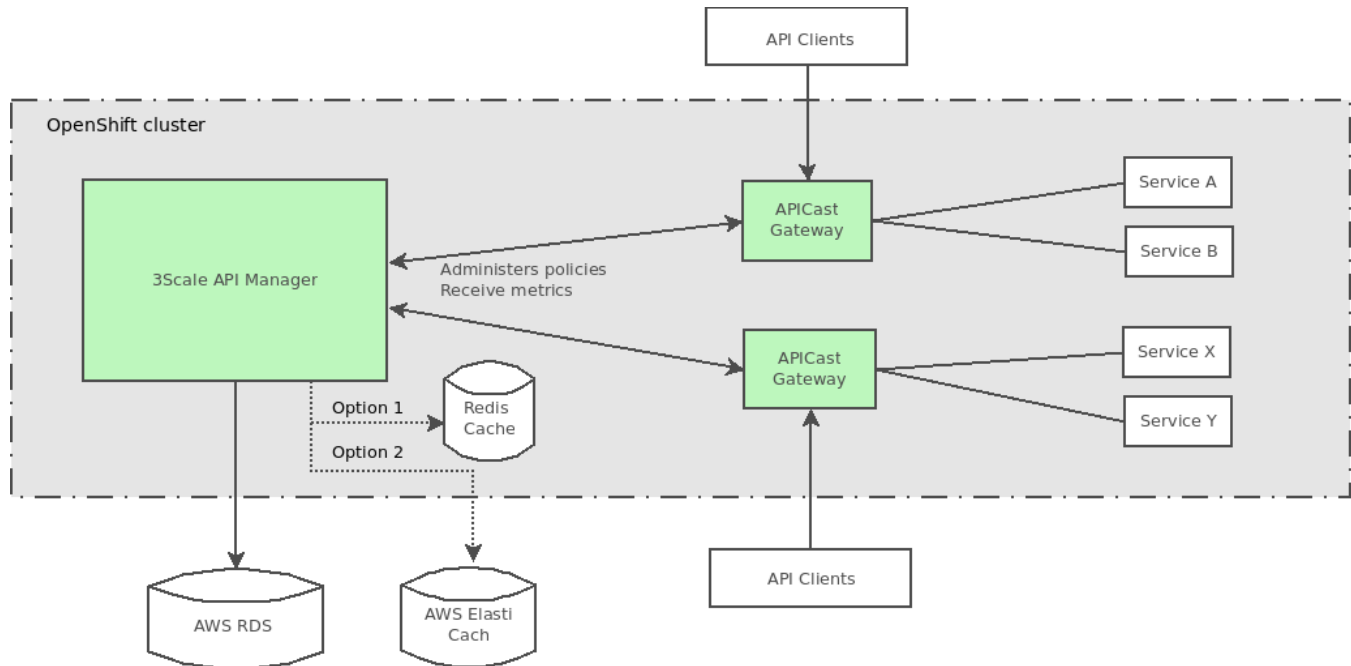
- Separate production and non-prod environments but with shared resources
- “Managed API Management Service” e.g. separate LoBs or projects within IRCC

Following provides a description of the roles and privileges at each level.

- Master Admin
 - Can create / delete tenants
 - Can edit details of tenants
 - Can impersonate tenants (gives access to all tenant’s API services and Admin Portal areas)
 - Aggregated view of tenants and analytics
- Tenant Admin (this is exactly the same as previously):
 - Can invite admin / members
 - Can manage admin / member rights and roles
 - Has access to all API services and Admin Portal areas
- Member:
 - Has access to whatever services / sections the admin has given them

All master admin and admin actions are available via UI and API.

3Scale Deployment Architecture



3Scale Production Recommendations

- Use an external database instead of the default database provided by the installation. We suggest using AWS RDS, which provides a fully managed database reducing operational overhead on the ESO team
- Use an external Redis cluster. Therefore are two options available.
 - You can use the Redis Operator available in the Operator Hub to set up and manage the Redis cluster. Red Hat provides support for this configuration and is limited to 3Scale only.
 - You can use AWS Elasti Cache. For more details and pricing - [read more](#).
- Please do not use the above Redis cluster for any application data.
- Consider using self managed APICast gateways instead of the built in API gateway.

As illustrated in the diagram above it allows multiple APICast gateways to segregate API products for scale, performance and security. The gateway can be placed in the same namespace as the services to ensure they are not accessible outside that namespace.

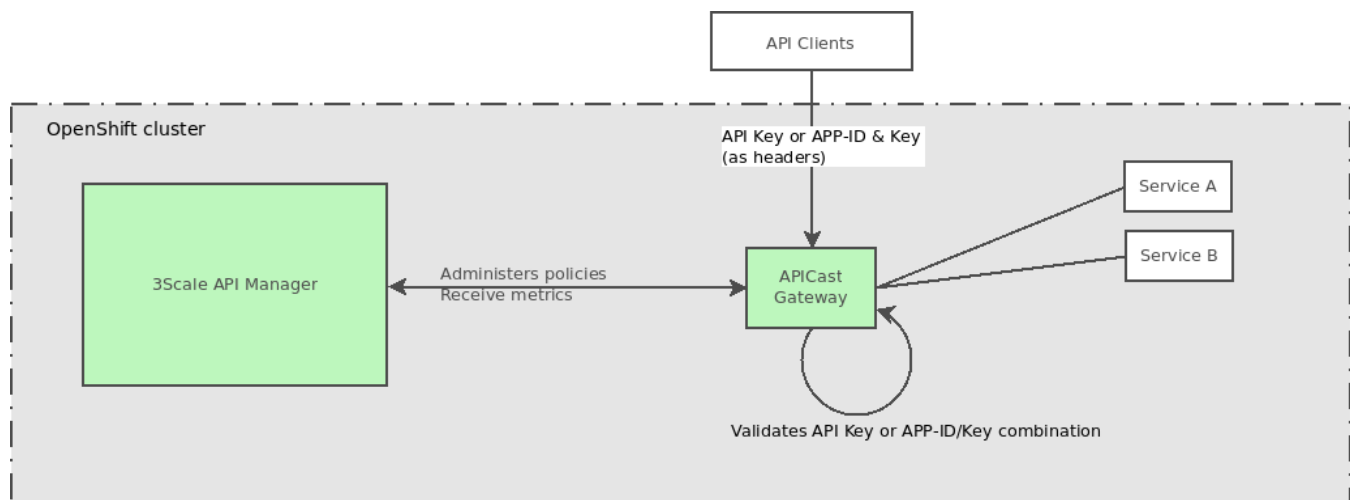
3 Scale API Authentication

The following table summarizes the options and the diagrams shows how it works in practice.



Authentication Mechanism	Description
API Key	Simple unique key used as header or query param
APP_ID & App_Key	The difference compared to above is while the APP_ID can remain the same, the App_Key can be rotated without downtime.
OpenID Connect	The API Clients may need significant rework to support OIDC based authentication. The token is validated using the provider public key and the JWT claim validation policy can provide extra validations

API Key / APP_ID & App_Key based Authenticaiton



OIDC Based Authentication

