IMPROV-Control Documentation

￼**￼**

Contents

[IMPROV Introduction 2](#_Toc108800776)

[IMPROV Overview 2](#_Toc108800777)

[Vlocity and IMPROV 2](#_Toc108800778)

[IMPROV 2.0 4](#_Toc108800779)

[Changes from IMPROV 2.0 to IMPROV 3.0 4](#_Toc108800780)

[IMPROV 3.0 Architecture 7](#_Toc108800781)

[IMPROV 3.0 Architecture, Brightspeed Specific 8](#_Toc108800782)

[ACN team's role to IMPROV 3.0 implementation 8](#_Toc108800783)

[IMPROV Control Changes and Bug Fixes 9](#_Toc108800784)

[Springboot Vulnerability Fix for Improv 9](#_Toc108800785)

[Improv-Control Code Update 9](#_Toc108800786)

[IMPROV-Control Deployment to Brightspeed GCP 11](#_Toc108800787)

[Set Up Source Repository in GCP 11](#_Toc108800788)

[Set Up Triggers for CloudBuild 11](#_Toc108800789)

[Creation of Secret Key and Certification 12](#_Toc108800790)

[Creation of API key 13](#_Toc108800791)

[Modify CloudBuild Configurations 14](#_Toc108800792)

[Sonarqube Configurations 18](#_Toc108800793)

[Deployment File 18](#_Toc108800794)

[Brightspeed GCP Deployment - Test Environment 20](#_Toc108800795)

[IMPROV-Control to IMPROV-Comms Integration 21](#_Toc108800796)

[IMPROV 3.0 APIs 21](#_Toc108800797)

[IMPROV CONTROL APIs 21](#_Toc108800798)

[IMPROV COMMS APIs 22](#_Toc108800799)

[Steps for requesting Connection to Lumen External Apigee 23](#_Toc108800800)

[IMPROV-Control Validation 23](#_Toc108800801)

[IMPROV Validation using Postman 23](#_Toc108800802)

# IMPROV Introduction

## IMPROV Overview

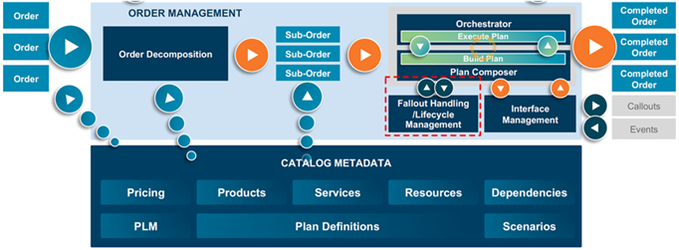
IMPROV is part of the Order to Cash, Buy-Flow process at Lumen. The application is triggered by Vlocity during its order management processes.​

When customers want to order service, they do it in Salesforce and then Salesforce communicates with Vlocity. Vlocity gathers all the information needed to be sent to the customers equipment. Vlocity uses IMPROV to send the data to the customers equipment. IMPROV's role is to collect data from Vlocity then transmit it along the way to the customer equipment with validation, call to LDAP, etc.​

## Vlocity and IMPROV

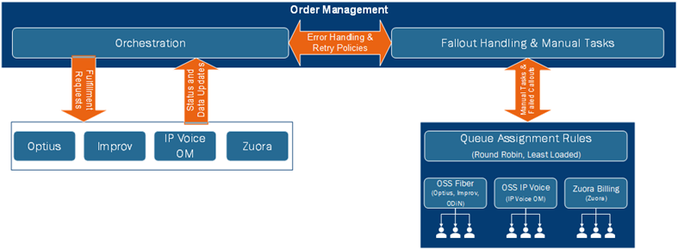
As mentioned in the introduction, Vlocity is the one who triggers IMPROV during its Order Management process, specifically during the Orchestration Process.

Image below shows the Order Management Process of Vlocity:



IMPROV is part of the Order Submit Flow before provisioning because it handles creation of CPE Credentials in the process.

Aside from the order submit flow, it is more often triggered during Order Fallout Handling, or what we call Unhappy Scenarios.



Here is a list of Scenario’s where Vlocity triggers IMPROV during its process flow

|  |  |  |
| --- | --- | --- |
| **Process** | **Part/Scenario** | **Flow** |
| New Install BuyFlow | N/A | N/A |
| Order Submit | Before Provisioning | [Apigee/GCP] Send Required details to IMPROV -> [IMPROV] Create CPE Credentials -> [Apigee/GCP] Handles the Response, and then provision |
| New Install -Tech Install | N/A | N/A |
| Unhappy Scenarios | Cancel Order before Activation | [Apigee/GCP] Initiate Cancelation of HSI -> [Apigee/GCP] Send Delete to IMPROV for HSI -> [IMPROV] Delete HSI Account |
|  | Active Customer Changes Product | [Apigee/GCP] Initiate upgrade/Downgrade APIs -> [Apigee/GCP] Update Speed Profile IMPROV -> [IMPROV] Update New HSI Speed |
|  | Suspend due to non-pay | [Apigee/GCP] Decompose suspend calls to OSS Systems -> [Apigee/GCP] Request Suspend to IMPROV -> [IMPROV] HSI set to suspend state |
|  | Resume due to non-pay | [Apigee/GCP] Decompose resume calls to OSS Systems -> [Apigee/GCP] Request Resume to IMPROV -> [IMPROV] HSI set to resume state |
|  | Disconnect | [Apigee/GCP] Initiate disconnect of HSI -> Send Delete to IMPROV for HSI -> [IMPROV] Delete HSI Account |

## IMPROV 2.0

IMPROV 2.0 is the current IMPROV Application in Lumen. it is written using Java 8 using springboot 2.2.5 Web. For more information regarding IMPROV 2.0, please refer to the document below:

[Improv 2.0 Solutions & Architecture.pptx (accenture.com)](https://ts.accenture.com/:p:/r/sites/CenturyLink-DFWDesignStudio/_layouts/15/Doc.aspx?action=edit&sourcedoc=%7Bf403e605-7aed-4b28-af3e-3b39bce52843%7D&wdOrigin=TEAMS-ELECTRON.recent.recent&wdExp=TEAMS-CONTROL)

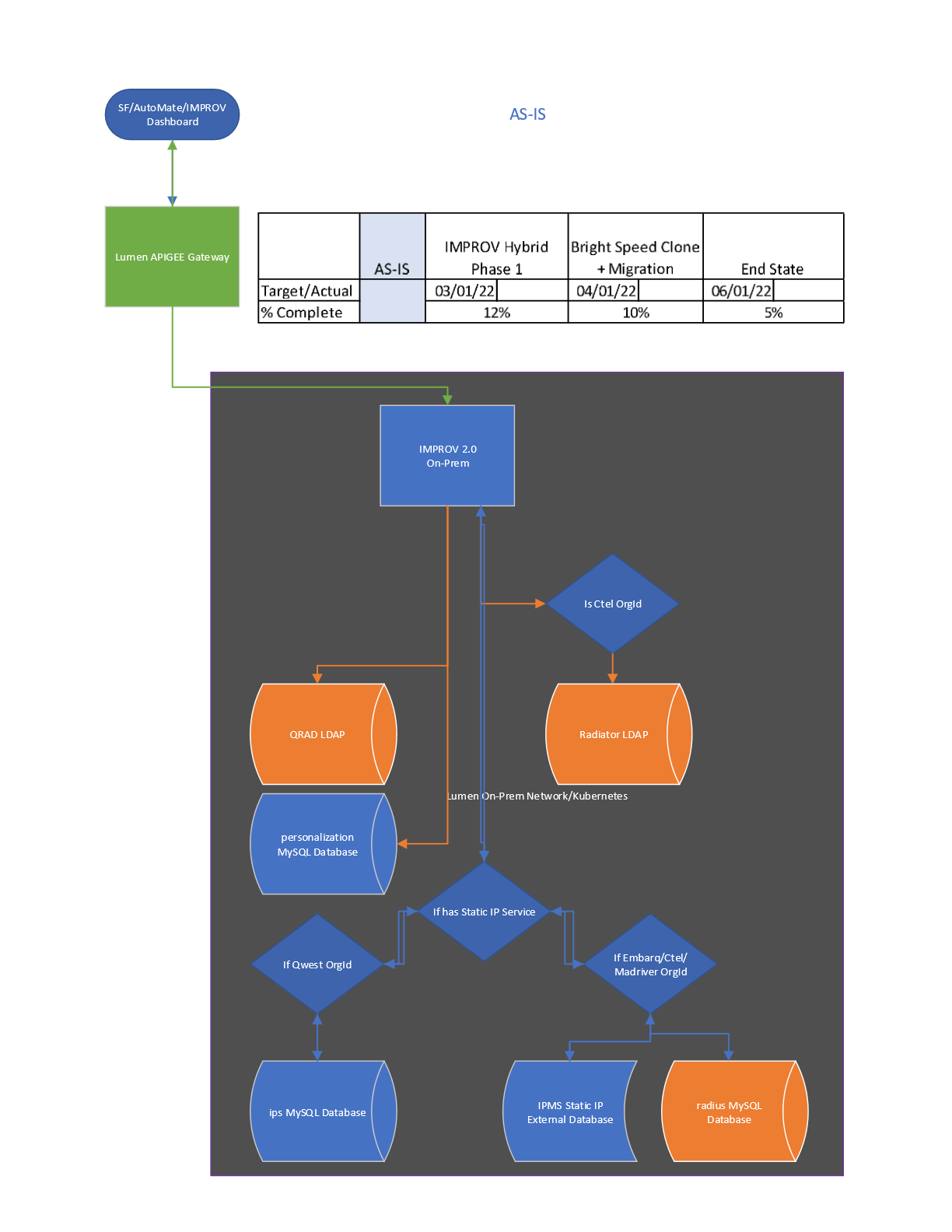
## Changes from IMPROV 2.0 to IMPROV 3.0

From the previous implementation, IMPROV 2.0 resides in the Lumen Network, part of OSS.

For IMPROV 3.0, it will be split into two parts: the IMPROV 3.0 Controller (resides in the Cloud) and IMPROV 3.0 Comm (resides in the Lumen Network)

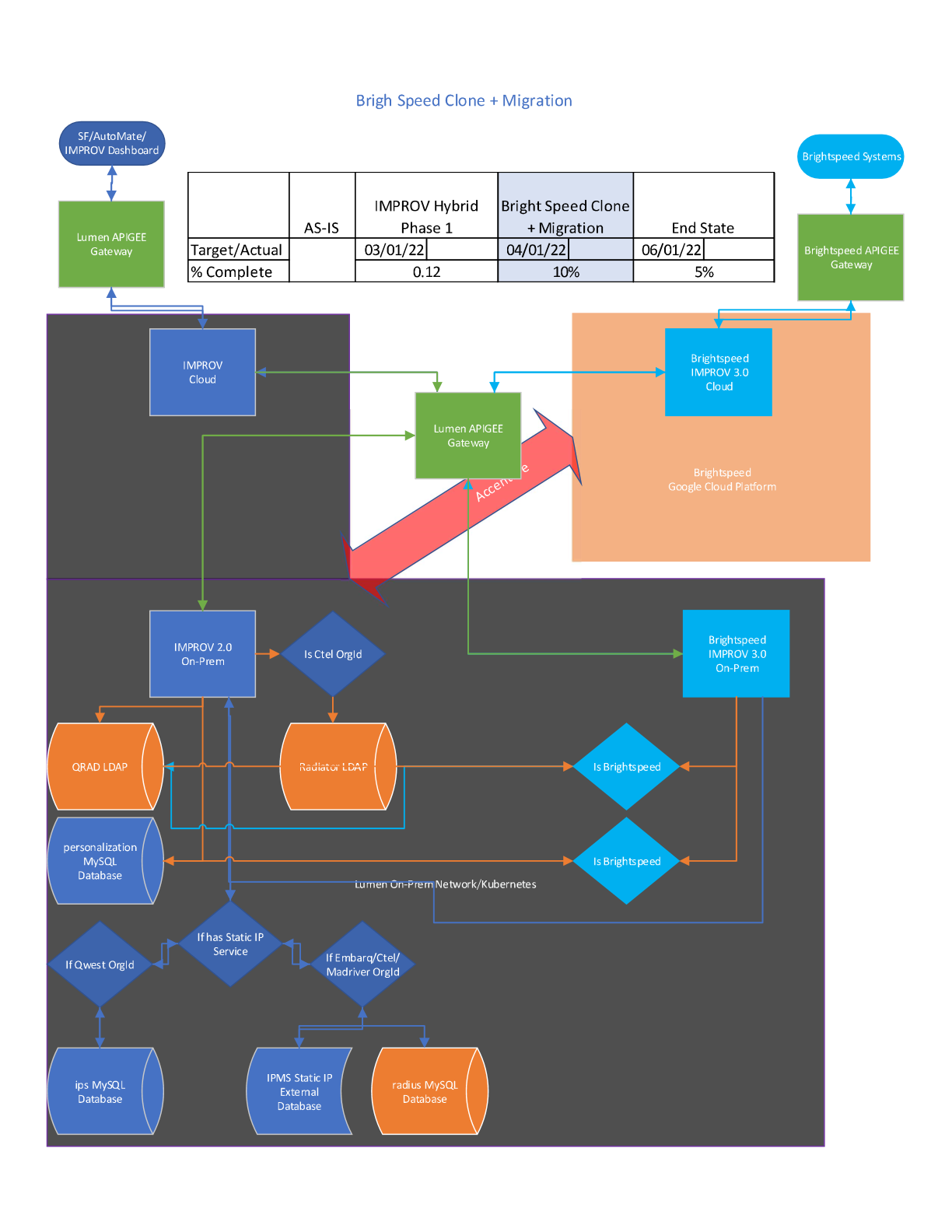
Comm’s job is to Communicate to the LDAP part of Lumen Network, while the Controller keeps the other functionalities of the previous implementation.

**IMPROV 2.0 Architecture**



link to visio: [IMPROV AS-IS.vsdx (accenture.com)](https://ts.accenture.com/:u:/r/sites/CenturyLink-DFWDesignStudio/_layouts/15/Doc.aspx?sourcedoc=%7BD48ED596-42C9-416B-A9B9-156630A17A0F%7D&file=IMPROV%20AS-IS.vsdx&action=default&mobileredirect=true&cid=b54f9152-ca56-48ce-a3d4-d48040c6484f)

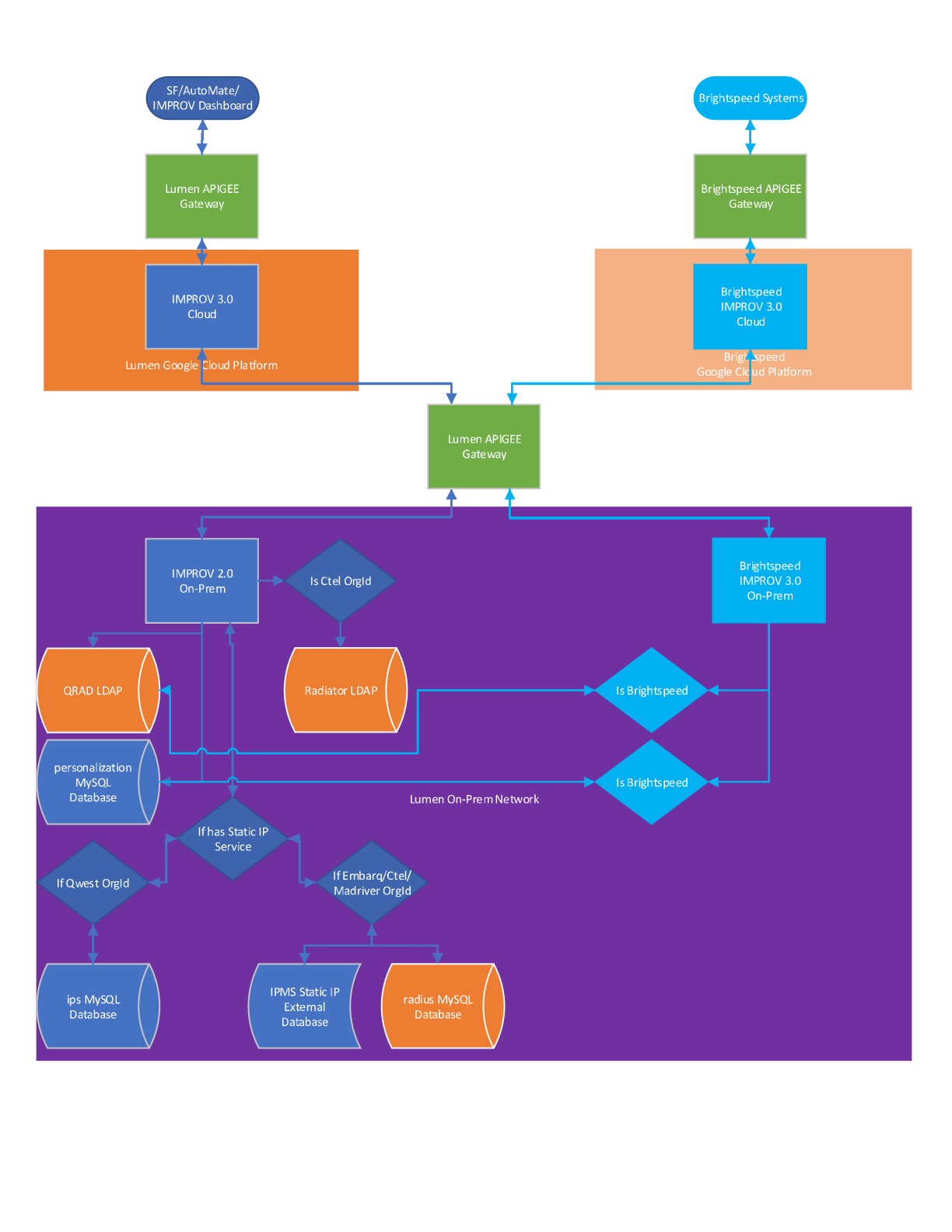
**IMPROV 2.0 Re-architecture Plan:**



link to visio: [IMPROV-BrightSpeed-Clone.vsdx (accenture.com)](https://ts.accenture.com/:u:/r/sites/CenturyLink-DFWDesignStudio/_layouts/15/Doc.aspx?sourcedoc=%7B41AE7600-77AF-474D-8732-6EE9740BFA54%7D&file=IMPROV-BrightSpeed-Clone.vsdx&action=default&mobileredirect=true)

## IMPROV 3.0 Architecture

After the re-architecture Plan, here is the new architecture plan for IMPROV 3.0:

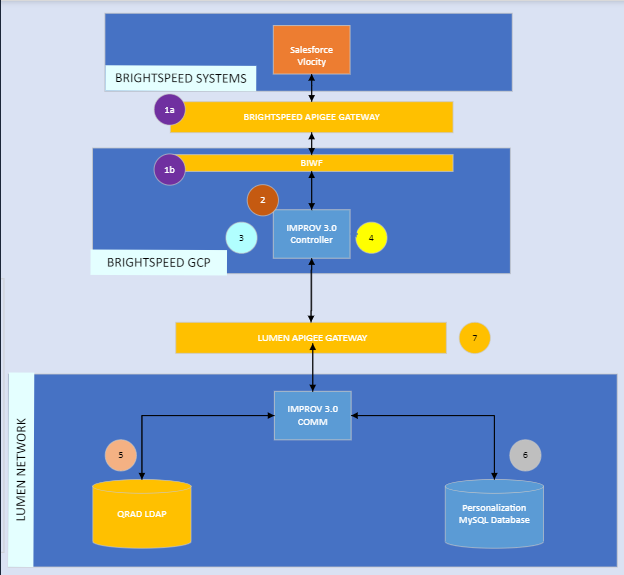


link to visio: [IMPROV 2.0 Flow Diagram V3.vsdx (accenture.com)](https://ts.accenture.com/:u:/r/sites/CenturyLink-DFWDesignStudio/_layouts/15/Doc.aspx?sourcedoc=%7B7E7CC777-59DA-4B95-B2E3-8C9DA4A62E05%7D&file=IMPROV%202.0%20Flow%20Diagram%20V3.vsdx&action=default&mobileredirect=true)

The split is planned to be implemented to both Lumen and Brightspeed systems, but as of the moment, Lumen won’t be able to implement the split on their side, thus making Brightspeed the first one to implement this change.

## IMPROV 3.0 Architecture, Brightspeed Specific

For a closer look of the IMPROV implementation in brightspeed, please refer to the diagram below:



For the details regarding each part of this diagram, please refer to this visio file: [IMPROV 3.0 Architecture Diagram (Brightspeed Specific).vsdx (accenture.com)](https://ts.accenture.com/:u:/r/sites/CenturyLink-DFWDesignStudio/_layouts/15/Doc.aspx?action=edit&sourcedoc=%7Bed2cb74e-9137-4e37-8065-25da7d68fc19%7D&or=PrevEdit)

## ACN team's role to IMPROV 3.0 implementation

For IMPROV 3.0, Changes in the code is implemented by the Lumen Development Team.

Lumen Team provided us the code for IMPROV Control to be deployed to GCP, while IMPROV-Comms will remain on their side, deployed to Lumen Network.

For the Accenture team, these are the summary of tasks:

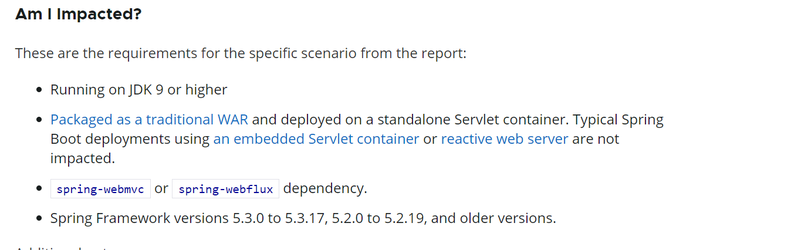
* Copy the code for IMPROV Control to Brightspeed Repository.
* Deploy IMPROV Control to GCP
* Verify that the deployment is working properly (Check APIs if reachable)
* provide endpoints to teams which has IMPROV as external dependency.
* provide IP to Lumen Team to for the to whitelist the IP we are using.
* Verify that we are able to connect to IMPROV Comms.

# IMPROV Control Changes and Bug Fixes

## Springboot Vulnerability Fix for Improv

Improv-control is not impacted on the current Springboot vulnerability. Improv is currently using Java 8. Action may not be necessary.

reference: [Spring Framework RCE, Early Announcement](https://spring.io/blog/2022/03/31/spring-framework-rce-early-announcement#suggested-workarounds)



## Improv-Control Code Update

**Additional Code**

There were changes on how Improv-Control will be integrated to Improv-Comms. Improv-Control has to repeatedly request for Access Token to Lumen API Marketplace every 30 minutes to connect with the endpoint of Improv-Comms. Because of this changes, this scheduled request of Access token is necessary to be added in the code.

Two classes were added in the improv-rules module:

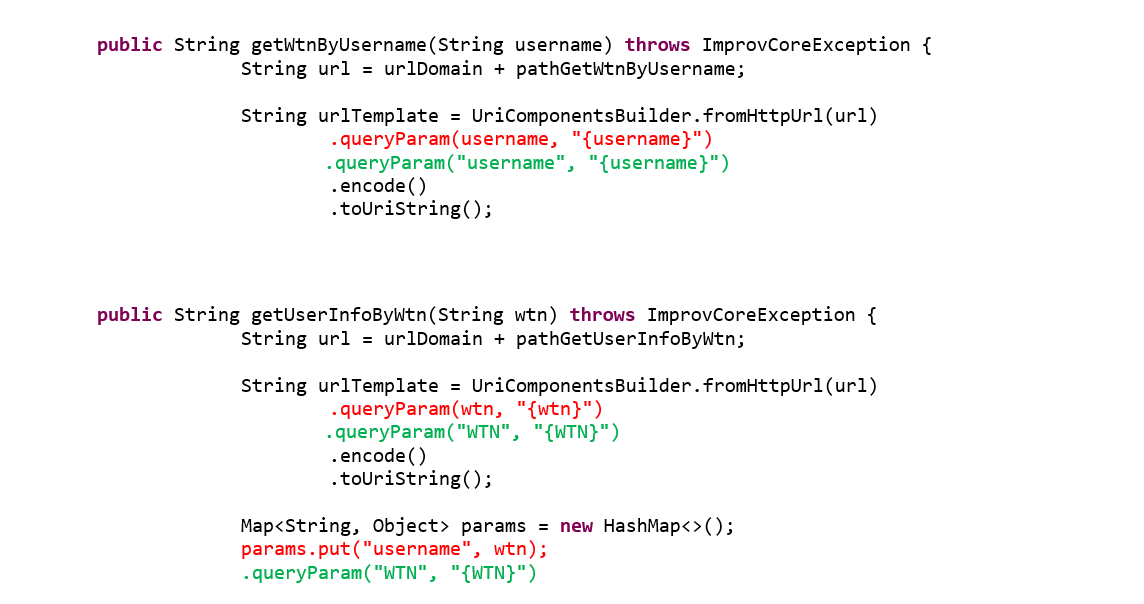
1. ImprovAuthentication.java
2. ImprovAuthRequest.java

**Code changes**

* **ImprovClient.java**

On the process of validating API calls, two paths were tested not working. In troubleshooting these problems, there were line of codes that need to be fixed.

In the ImprovClient.java file,



lines in red were replaced with lines in green. These changes resolved the error in the API call for **getUserInfoByWtn** and **getWtnByUsername**.

* **application.properties**

We updated the values of some properties to match on the sandbox environment of Lumen instance.

cxg7.jwt.claim.iss

cxg7.jwt.claim.sub

cxg7.jwt.pubkey

improv.api.username

improv.api.password

improv.api.oAuthToken

improv.api.consumerKey

improv.api.consumerSecret

improv.api.tokenEndpoint

improv.api.grantType

improv.api.url.domain

improv.api.url.path.getOrg

improv.api.url.path.platformClassMap

improv.api.url.path.classServiceAttrMap

improv.api.url.path.findUser

improv.api.url.path.getWtnByUsername

improv.api.url.path.getUserInfoByWtn

improv.api.url.path.createAccount

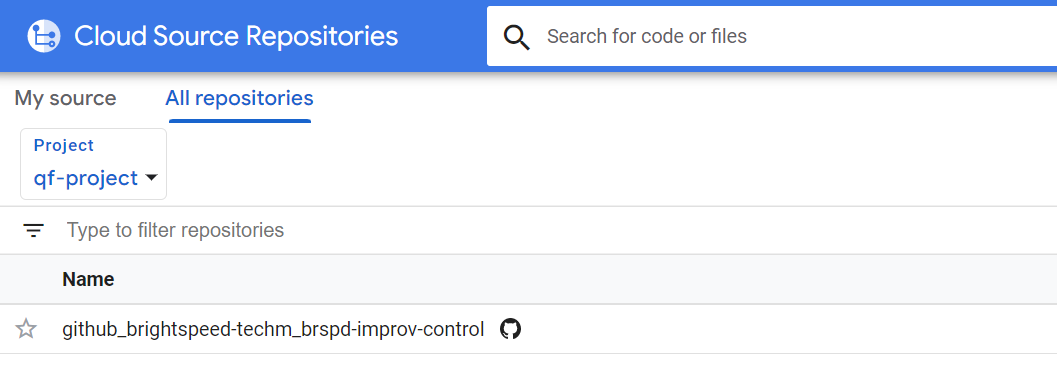
improv.api.url.path.updateUserAccount

improv.api.url.path.deleteUserAccount

improv.api.url.path.enableUserAccount

# IMPROV-Control Deployment to Brightspeed GCP

## Set Up Source Repository in GCP

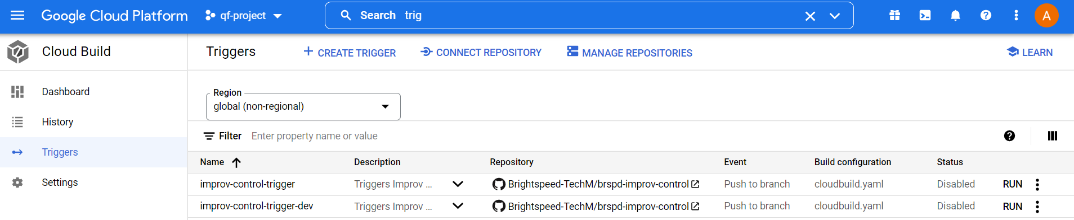


This Source Repository for improv (**github\_brightspeed-techm\_brspd-improv-control**) was used by dev in running Improv-cloud app through cloud shell.

These are the actions performed to setup the repository:

1. Ensure first that Github repository of Brightspeed permits the connection to GCP project.
   1. Github Org: brightspeed
   2. repo name: brspd-improv-control
2. Request for Source Repository Administrator role to manage Source repositories.
3. In **qf-project**, search for Source Repository.
4. Click Add repository.
5. Select “Connect external repository” to mirror the Github repository.
6. Select **qf-project** for Project and **Github** as Git provider.
7. Check I authorize Google Cloud platform project to store third-party authentication credentials
8. Click Connect to Github.

## Set Up Triggers for CloudBuild



We created triggers that will automatically start our cloud build whenever we push or make changes to our source repository branch.

[**improv-control-master-trigger**](https://console.cloud.google.com/cloud-build/triggers;region=global/edit/00c21ec2-bcd2-4c5d-a299-1fadbae000b7?project=qf-project) : master branch

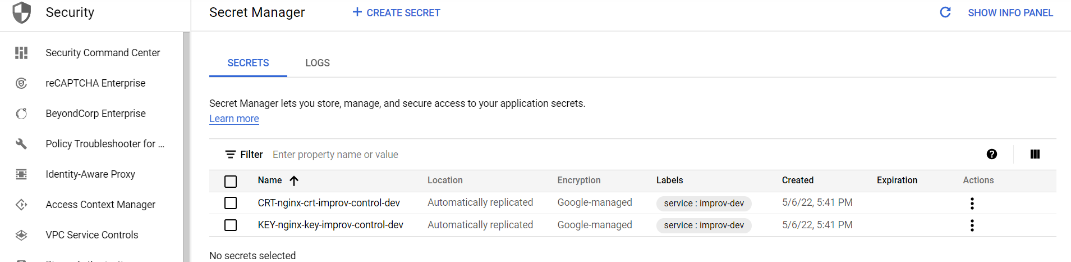
**improv-control-trigger-dev** : improv-control-dev-branch

These triggers are connected to **Brightspeed-TechM/brspd-improv-control** repository from Github and will run the cloud build configuration file (cloudbuild.yaml).

Our trigger was created following these steps:

1. In **qf-project,** search for Cloud Build.
2. In the Cloud Build page, click Triggers.
3. Click create trigger.
4. Provide a unique name within the project’s region in the name field.
5. Under event, select “Push to a branch” so that trigger will start the build whenever there is a push event in a branch of the repository.
6. Select a source repository. Connect to a new one if there’s no existing repository.
7. Select which branch of the repository will be detected with push event.
8. Under Configuration>Type, select “Cloud Build configuration file (yaml or json)”. We created the build steps in a yaml file.
9. Under Configuration>Location, choose Repository where the configuration file is also located and provide the file path in the Cloud Build configuration file location field.
10. On Substitution variables > Click add variables
    1. \_TARGET\_PROJECT : qf-project
    2. \_ENV : dev
    3. \_VERSION : 1.0.0
    4. \_CLUSTER\_NAME : improv-control-dev-pvt-cluster-1
    5. \_REGION : us-east1-b
11. Click Create.

## Creation of Secret Key and Certification



**Run this command in Cloud shell of your project:**

openssl req -x509 -nodes -days 365 -newkey rsa:2048 \  
-keyout ./nginx.key -out ./nginx.crt

Note: On COMMON - NAME = brspd-improv-control-dev.endpoints.qf-project.cloud.goog

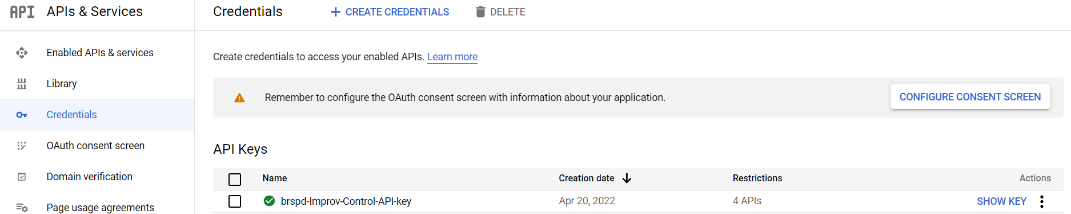
(You just skipped other properties beside common name)

after

gcloud secrets create CRT-nginx-crt-improv-control-dev --data-file=nginx.crt --labels=service=improv-dev  
gcloud secrets create KEY-nginx-key-improv-control-dev --data-file=nginx.key --labels=service=improv-dev

setting and connection of key and certificate to Improv service are included in the cloud build steps

## Creation of API key



APIs of improv control are protected by API key. External services accessing the APIs are required to provide the API key in sending request

API key for Improv-control is setup following these steps:

1. Under **qf-project**, Open APIs & Services>Credentials
2. Click Create credentials and choose API key.
3. A window with an API Key will be shown. Click close.
4. A list of API keys is shown in the page. Click the newly created key.
5. Rename the key (brspd-Improv-Control-API-key).
6. Under API restrictions, click restrict key and select which enabled APIs (Improv API) that this key can call.
7. Click Save.
8. This API key is given to other projects/app that will access Improv.

## Modify CloudBuild Configurations

Improv-control is deployed through Cloud Build. Build steps were configured in **cloudbuild.yaml**. A trigger **(improv-control-trigger-dev)** was created in GCP to automatically run these steps configured in this file.

cloudbuild.yaml contains the following steps/scripts:

steps:

- name: 'gcr.io/google.com/cloudsdktool/cloud-sdk'

id: 'Connect to a project'

entrypoint: 'gcloud'

args:

- 'config'

- 'set'

- 'project'

- '$\_TARGET\_PROJECT'

- name: 'gcr.io/google.com/cloudsdktool/cloud-sdk'

id: 'Connect to GKE Cluster'

entrypoint: 'gcloud'

args:

- 'container'

- 'clusters'

- 'get-credentials'

- '$\_CLUSTER\_NAME'

# - '--region=$\_REGION'

- '--zone=$\_ZONE'

- '--project=$\_TARGET\_PROJECT'

# One time

- name: 'gcr.io/google.com/cloudsdktool/cloud-sdk'

id: 'Deploy IMPROV API'

entrypoint: gcloud

args:

- 'endpoints'

- 'services'

- 'deploy'

- 'improv-schema/src/main/resources/open-api-specs/brspd-improv-control-openapi-$\_ENV.yaml'

# One time

- name: 'gcr.io/google.com/cloudsdktool/cloud-sdk'

id: 'Enable IMPROV API'

entrypoint: gcloud

args:

- 'services'

- 'enable'

- 'brspd-improv-control-$\_ENV.endpoints.$\_TARGET\_PROJECT.cloud.goog'

#

- name: maven:3.6.3-jdk-8

id: 'Clean Install improv resources'

entrypoint: 'mvn'

args: ['clean', 'install', '-DskipTests']

- name: maven:3.6.0-jdk-11-slim

entrypoint: 'mvn'

id: 'Sonarqube Build'

args: ['sonar:sonar','-Dsonar.projectKey=improv-cloud', '-Dsonar.host.url=http://34.138.202.107:9121/', '-Dsonar.login=e13b1a9a6d07d5944196e69a2224cbb8e4792c36']

- name: 'gcr.io/cloud-builders/docker'

id: 'Build Docker Image'

args: ['build','-t','gcr.io/$\_TARGET\_PROJECT/improv-cloud-$\_ENV:$\_VERSION','.']

- name: 'gcr.io/cloud-builders/docker'

id: 'Docker Push'

args: ["push", "gcr.io/$\_TARGET\_PROJECT/improv-cloud-$\_ENV:$\_VERSION"]

# Authorize Cloud Build to Access the Private Cluster (Enable Control Plane Authorized Networks)

- name: 'gcr.io/google.com/cloudsdktool/cloud-sdk'

id: 'Authorize Cloud Build'

entrypoint: 'bash'

args:

- -c

- |

apt-get install dnsutils -y &&

cloudbuild\_external\_ip=$(dig @resolver4.opendns.com myip.opendns.com +short) &&

gcloud container clusters update $\_CLUSTER\_NAME '--zone=$\_ZONE' --enable-master-authorized-networks --master-authorized-networks $cloudbuild\_external\_ip/32 &&

echo $cloudbuild\_external\_ip

- name: 'gcr.io/google.com/cloudsdktool/cloud-sdk'

id: 'Configure kubectl'

entrypoint: gcloud

args:

- 'container'

- 'clusters'

- 'get-credentials'

- '$\_CLUSTER\_NAME'

# - '--region=$\_REGION'

- '--zone=$\_ZONE'

- name: 'gcr.io/google.com/cloudsdktool/cloud-sdk'

id: 'List Pods'

entrypoint: kubectl

args:

- 'get'

- 'pods'

- name: 'gcr.io/google.com/cloudsdktool/cloud-sdk'

id: 'Apply namespace'

entrypoint: kubectl

args:

- 'apply'

- '--filename=./namespace.yaml'

# Preparing secrets from Secret Manager

- name: 'gcr.io/google.com/cloudsdktool/cloud-sdk'

id: 'Access and Prepare Secrets from Secret Manager'

entrypoint: 'bash'

args:

- -c

- |

gcloud secrets versions access latest --secret=CRT-nginx-crt-improv-control-$\_ENV --format='get(payload.data)' | tr '\_-' '/+' | base64 -d > nginx.crt &&

gcloud secrets versions access latest --secret=KEY-nginx-key-improv-control-$\_ENV --format='get(payload.data)' | tr '\_-' '/+' | base64 -d > nginx.key

# Creating GKE secret (NGINX Cert and Key) that belongs to the namespace

- name: 'gcr.io/google.com/cloudsdktool/cloud-sdk'

id: 'Configure Secret - NGINX Cert & Key'

entrypoint: bash

args:

- -c

- |

export secrets=$(kubectl get secrets -o name -n improv-cloud-namespace )

if [[ $secrets != \*"secret/nginx-ssl"\* ]]

then

kubectl create secret generic nginx-ssl --from-file=nginx.crt --from-file=nginx.key -n improv-cloud-namespace

else

echo "Secret nginx-ssl exists."

fi

# Update variable in deployment file

- name: 'gcr.io/google.com/cloudsdktool/cloud-sdk'

id: 'Update IMAGE\_SOURCE in deployment file'

entrypoint: 'bash'

args:

- -c

- |

image\_name=gcr.io/$\_TARGET\_PROJECT/improv-cloud-$\_ENV:$\_VERSION &&

sed -i s@IMAGE\_SOURCE@$image\_name@g ./deployment-files/$\_ENV/deployment.yaml &&

sed -i s@GCP-PROJECT-ID@$\_TARGET\_PROJECT@g ./deployment-files/$\_ENV/deployment.yaml

- name: 'gcr.io/google.com/cloudsdktool/cloud-sdk'

id: 'Apply deployment'

entrypoint: kubectl

args:

- 'apply'

- '--filename=./deployment-files/$\_ENV/deployment.yaml'

- '--namespace=improv-cloud-namespace'

- name: 'gcr.io/google.com/cloudsdktool/cloud-sdk'

id: 'Expose service'

entrypoint: bash

args:

- -c

- |

export services=$(kubectl get services -o name -n improv-cloud-namespace)

if [[ $services == \*"service/improv-cloud-service-$\_ENV"\* ]]

then

kubectl delete service improv-cloud-service-$\_ENV -n improv-cloud-namespace

kubectl apply --filename=./deployment-files/$\_ENV/service.yaml --namespace=improv-cloud-namespace

else

kubectl apply --filename=./deployment-files/$\_ENV/service.yaml --namespace=improv-cloud-namespace

fi

# Provisioning of external load balancer takes about 20-40seconds

- name: 'gcr.io/google.com/cloudsdktool/cloud-sdk'

id: 'Wait Time of 1.5-minute'

entrypoint: 'bash'

args: ['-c', 'sleep 90s']

# Getting and updating the IP Address

- name: 'gcr.io/google.com/cloudsdktool/cloud-sdk'

id: 'Update IP Address on YAML'

entrypoint: 'bash'

args:

- -c

- |

my\_external\_ip=$(kubectl -n improv-cloud-namespace get svc improv-cloud-service-$\_ENV -o jsonpath="{.status.loadBalancer.ingress[\*].ip}") &&

sed -i s/111.111.111.111/$my\_external\_ip/ improv-schema/src/main/resources/open-api-specs/brspd-improv-control-openapi-$\_ENV.yaml &&

echo $my\_external\_ip

# Redeploying the endpoint with the updated manifest (YAML) file

- name: 'gcr.io/google.com/cloudsdktool/cloud-sdk'

id: 'Re-deploy IMPROV Endpoints to Target Correct IP'

entrypoint: gcloud

args:

- 'endpoints'

- 'services'

- 'deploy'

- 'improv-schema/src/main/resources/open-api-specs/brspd-improv-control-openapi-$\_ENV.yaml'

# Disable Control Plane Authorized Networks after Deployment

- name: 'gcr.io/google.com/cloudsdktool/cloud-sdk'

id: 'Disable Authorized Networks'

entrypoint: 'gcloud'

args:

- 'container'

- 'clusters'

- 'update'

- '$\_CLUSTER\_NAME'

# - '--region=$\_REGION'

- '--zone=$\_ZONE'

- '--no-enable-master-authorized-networks'

images: ['gcr.io/$\_TARGET\_PROJECT/improv-cloud-$\_ENV:$\_VERSION']

timeout: 900s

## Sonarqube Configurations

Improv-Control code is automatically analyze by Sonarqube through CloudBuild. The codes are analyzed using default quality profile.

Brightspeed Sonarqube instance:

[http://34.138.202.107:9121/sessions/new?return\_to=%2F](https://urldefense.proofpoint.com/v2/url?u=http-3A__34.138.202.107-3A9121_sessions_new-3Freturn-5Fto-3D-252F&d=DwMFAg&c=eIGjsITfXP_y-DLLX0uEHXJvU8nOHrUK8IrwNKOtkVU&r=CWCGi_tXG1ocQkEQLSZP-c6pIxMRD8-Wm5cxZip9V_8&m=zs0hXEluwLkqtQnRKqSyIMk-oaefLAm0-X9Fn2QMwPImBIMLdXB-CXI-E8nFa8_V&s=5r2NrYeO05rvU_mB8CwozdBt9XO3h17DarrYqx_iMBQ&e=)

Project name: Improv-Parent

## Deployment File

Docker image of Improv-control is deployed in a container in qf-project using the scripts in **deployment.yaml.**

This also includes deploying the app in esp to expose service in a google endpoint (**brspd-improv-control-dev.endpoints.qf-project.cloud.goog**).

deployment yaml script:

apiVersion: apps/v1

kind: Deployment

metadata:

name: improv-cloud

namespace: improv-cloud-namespace

labels:

app: improv

spec:

replicas: 2

selector:

matchLabels:

app: improv

template:

metadata:

labels:

app: improv

spec:

volumes:

- name: nginx-ssl

secret:

secretName: nginx-ssl

containers:

# [START esp]

- name: esp

image: gcr.io/endpoints-release/endpoints-runtime:1

args: [

"--http\_port=8080",

"--ssl\_port=443",

"--backend=127.0.0.1:8081",

"--service=brspd-improv-control-dev.endpoints.qf-project.cloud.goog",

"--rollout\_strategy=managed",

]

# [END esp]

ports:

- containerPort: 8081

- containerPort: 443

volumeMounts:

- mountPath: /etc/nginx/ssl

name: nginx-ssl

readOnly: true

- name: improv-cloud

image: IMAGE\_SOURCE

env:

- name: GCLOUD\_PROJECT

value: GCP-PROJECT-ID

Improv-Control service is exposed using an Internal Load Balancer in Brightspeed GCP. Scripts in **service.yaml** were executed during the cloudbuild to successfully exposed it as a service.

service.yaml script

apiVersion: v1

kind: Service

metadata:

name: improv-cloud-service-dev

annotations:

networking.gke.io/load-balancer-type: "Internal"

networking.gke.io/internal-load-balancer-allow-global-access: "true"

spec:

ports:

# - port: 80

# targetPort: 8081

- port: 443

protocol: TCP

name: https

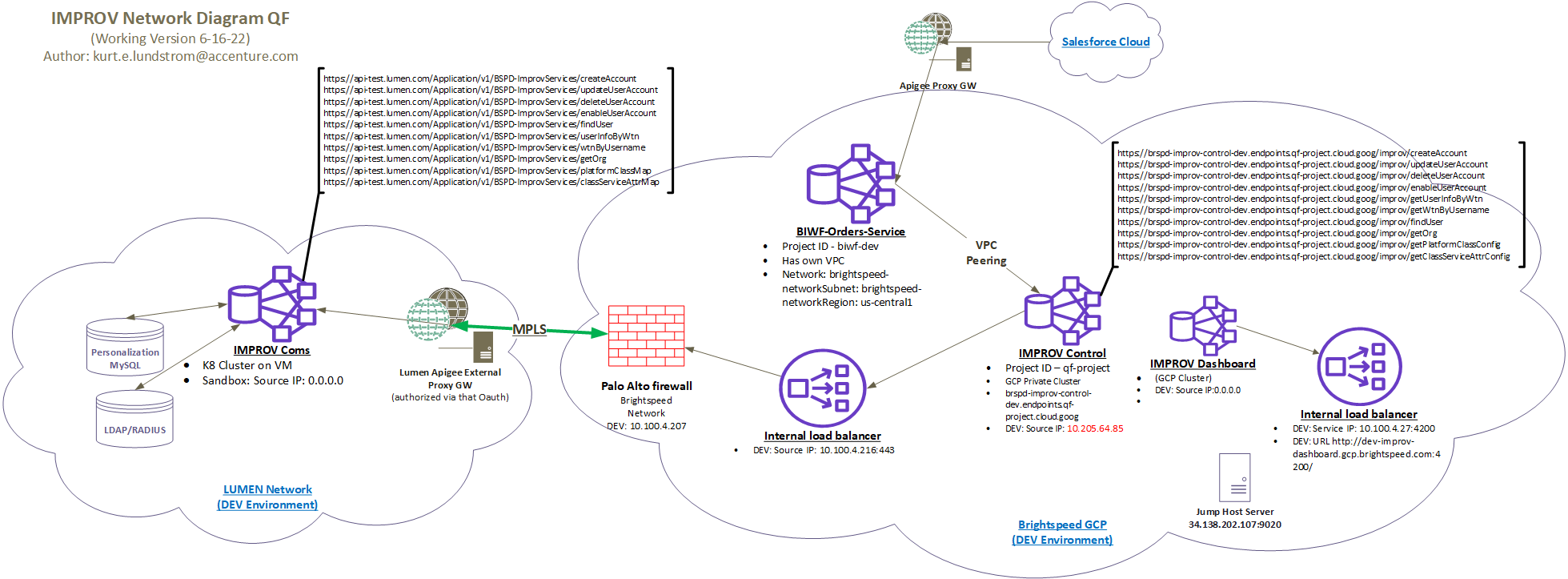
selector:

app: improv

type: LoadBalancer

## IMPROV Dev Network Diagram

Image Below is a visualization of IMPROV Network after deployment to the Dev environment:



## Brightspeed GCP Deployment - Test Environment

This document describes all the actions needed to successfully deploy Improv in Brightspeed GCP - test environment.

**Test Environment Preparation**

I. Enable GCP APIs and Services – You may review the existing APIs on the qf-project  
<https://console.cloud.google.com/apis/dashboard?referrer=search&project=qf-project>

II. Update the Service Account Roles

* [PROJECT-ID]@cloudbuild.gserviceaccount.com
* [PROJECT-ID]-compute@developer.gserviceaccount.com

III. Setup SECRET - brspd-improv-ctrl-test.endpoints.[project-id].cloud.goog

IV.Setup Cloud Build Trigger | Repository

V. Take note of ZONE vs. REGION

VI. Trigger Build

VII. Create API Key

VIII. Test API

**Details**

* Private Cluster
  + cluster name: bs-prj-improv-uat-01-improv-gke-pvt-01
  + project name: bs-prj-improv-uat-01
* Repo branch (Github)
  + branch name: master
* Create trigger
  + trigger name: improv-control-trigger-test
  + Cloud build file:  cloudbuild.yaml
  + Environment Variables:
    - \_TARGET\_PROJECT : bs-prj-improv-uat-01
    - \_ENV : TEST
    - \_VERSION : 1.0.0
    - \_CLUSTER\_NAME : bs-prj-improv-uat-01-improv-gke-pvt-01
    - \_REGION : us-east4
* Create API-key
  + name: brspd-Improv-Control-test-API-key
* Create secrets
  + run in cloud shell

openssl req -x509 -nodes -days 365 -newkey rsa:2048   
-keyout ./nginx.key -out ./nginx.crt

On COMMON - NAME = brspd-improv-ctrl-test.endpoints.bs-prj-improv-uat-01.cloud.goog

gcloud secrets create CRT-nginx-crt-improv-control-test --data-file=nginx.crt --labels=service=improv-test  
gcloud secrets create KEY-nginx-key-improv-control-test --data-file=nginx.key --labels=service=improv-test

* Update config/yaml files (for deployment)
  + deployment.yaml
  + service.yaml
  + cloudbuild.yaml
  + brspd-improv-control-openapi.yaml

**Deployment actions**

* Push to repo to trigger cloud build
* Automated creation of GCP components through cloud build
  + image, workloads,services(internal load balancer)
* test integrations (API calls)

# IMPROV-Control to IMPROV-Comms Integration

## IMPROV 3.0 APIs

IMPROV’s parts communicate with each other via URL endpoints which are used for HTTP requests. here are the list of URL endpoints:

### IMPROV CONTROL APIs

Improv-control APIs are exposed through the following URL:

**Authentication**: API key

|  |
| --- |
| <https://brspd-improv-control-dev.endpoints.qf-project.cloud.goog/improv/createAccount> |
| <https://brspd-improv-control-dev.endpoints.qf-project.cloud.goog/improv/updateUserAccount> |
| <https://brspd-improv-control-dev.endpoints.qf-project.cloud.goog/improv/deleteUserAccount> |
| <https://brspd-improv-control-dev.endpoints.qf-project.cloud.goog/improv/enableUserAccount> |
| <https://brspd-improv-control-dev.endpoints.qf-project.cloud.goog/improv/getUserInfoByWtn> |
| <https://brspd-improv-control-dev.endpoints.qf-project.cloud.goog/improv/getWtnByUsername> |
| <https://brspd-improv-control-dev.endpoints.qf-project.cloud.goog/improv/findUser> |
| <https://brspd-improv-control-dev.endpoints.qf-project.cloud.goog/improv/getOrg> |
| <https://brspd-improv-control-dev.endpoints.qf-project.cloud.goog/improv/getPlatformClassConfig> |
| <https://brspd-improv-control-dev.endpoints.qf-project.cloud.goog/improv/getClassServiceAttrConfig> |

### IMPROV COMMS APIs

Comms Endpoints, entry point of improv-control via Lumen Apigee

**Authentication**: OAuth

|  |
| --- |
| <https://api-test.lumen.com/Application/v1/BSPD-ImprovServices/createAccount> |
| <https://api-test.lumen.com/Application/v1/BSPD-ImprovServices/updateUserAccount> |
| <https://api-test.lumen.com/Application/v1/BSPD-ImprovServices/deleteUserAccount> |
| <https://api-test.lumen.com/Application/v1/BSPD-ImprovServices/enableUserAccount> |
| <https://api-test.lumen.com/Application/v1/BSPD-ImprovServices/userInfoByWtn> |
| <https://api-test.lumen.com/Application/v1/BSPD-ImprovServices/wtnByUsername> |
| <https://api-test.lumen.com/Application/v1/BSPD-ImprovServices/findUser> |
| <https://api-test.lumen.com/Application/v1/BSPD-ImprovServices/getOrg> |
| <https://api-test.lumen.com/Application/v1/BSPD-ImprovServices/getPlatformClassConfig> |
| <https://api-test.lumen.com/Application/v1/BSPD-ImprovServices/getClassServiceAttrConfig> |

## Steps for requesting Connection to Lumen External Apigee

To Connect the GCP Application to Lumen Apigee Gateway, We need to set up the application to their API Marketplace. after setting this up, It will provide consumer key and consumer secret, which will be used to generate the oAuth token which is the authentication when passing through the gateway.

Note that this oAuth token expires every 30 mins, so we embedded the steps for generation of oAuth Token in the improv-control code, and make this variable updated every 30 mins as well.

For the steps, refer to this link:

[Using OAuth 2.0 to Access Lumen API's | Lumen API Marketplace](https://apimarketplace.lumen.com/using-oauth-20-access-lumen-apis)

More detailed link is also available, but can only be accessible inside Lumen Network:

<https://centurylink.sharepoint.com/sites/SPTAPICoP/SitePages/API-Security-Endpoint.aspx>

# IMPROV-Control Validation

## IMPROV Validation using Postman

We validate IMPROV URLs connectivity via Postman.

For an overview on how to use Postman, please refer to this link:

[How to Use Postman: The Fundamentals [Video] | POST/CON 2019 | Postman](https://www.postman.com/post-con-2019/how-to-use-postman/)

**We have these initial documents from Lumen Dev and Validation Team for us to guide us in our Validation:**

Postman Collection from Lumen’s Validator

[Lumen - Quantum Fiber - IMPROV.postman\_collection.json - Default (accenture.com)](https://ts.accenture.com/sites/CenturyLink-DFWDesignStudio/Shared%20Documents/Forms/Default.aspx?FolderCTID=0x0120007D83A8F2CDAF27449A4FEF1D6DE511DA&id=%2Fsites%2FCenturyLink%2DDFWDesignStudio%2FShared%20Documents%2FQF%20Stack%20Clone%2F02%2E%20Plan%20%26%20Build%20%28OSS%29%2FImprov%2FIMPROV%2Epostman%5Fcollection%2Ejson&parent=%2Fsites%2FCenturyLink%2DDFWDesignStudio%2FShared%20Documents%2FQF%20Stack%20Clone%2F02%2E%20Plan%20%26%20Build%20%28OSS%29%2FImprov)

Lumen’s API Documentation

[Lumen On-Prem - BSPD-IMPROV-SERVICES.docx (accenture.com)](https://ts.accenture.com/:w:/r/sites/CenturyLink-DFWDesignStudio/_layouts/15/Doc.aspx?sourcedoc=%7B399BA233-DF3A-4EE3-AD0D-1F6BC13530EA%7D&file=Lumen%20On-Prem%20-%20BSPD-IMPROV-SERVICES.docx&action=default&mobileredirect=true&cid=ebe0cefd-e0e1-445b-a585-1eeb35f631e2)

For our Validation, we Created a new collection in postman to test if the APIs, aligned with the documentation:

[Updated Improv Postman Collection](https://ts.accenture.com/sites/CenturyLink-DFWDesignStudio/Shared%20Documents/Forms/Default.aspx?FolderCTID=0x0120007D83A8F2CDAF27449A4FEF1D6DE511DA&id=%2Fsites%2FCenturyLink-DFWDesignStudio%2FShared%20Documents%2FQF%20Stack%20Clone%2F02%2E%20Plan%20%26%20Build%20%28OSS%29%2FImprov%2FIMPROV-Updated%2Epostman_collection&parent=%2Fsites%2FCenturyLink-DFWDesignStudio%2FShared%20Documents%2FQF%20Stack%20Clone%2F02%2E%20Plan%20%26%20Build%20%28OSS%29%2FImprov)

This is the list of APIs tested via postman and the status.

|  |  |
| --- | --- |
| API | STATUS |
| getPlatformClassConfig | Success |
| getClassServiceAttrConfig | Success |
| findUser\_username | Success |
| findUser\_WTN | Success |
| deleteUserAccount | Success |
| updateUserAccount | Success |
| createAccount | Success |
| getOrg | Success |
| getUserInfoByWtn | Success |
| getWtnByUsername | Success |
| enableUserAccount | Success |