Zero to Cloud-Native with IBM Cloud

Part 9C: Finish Deploying and Testing the Application

Kevin Collins

Technical Sales Leader

IBM Cloud Enterprise Containers – Americas

Kunal Malhotra

Cloud Platform Engineer

IBM Cloud MEA

1 Create remaining Pipelines

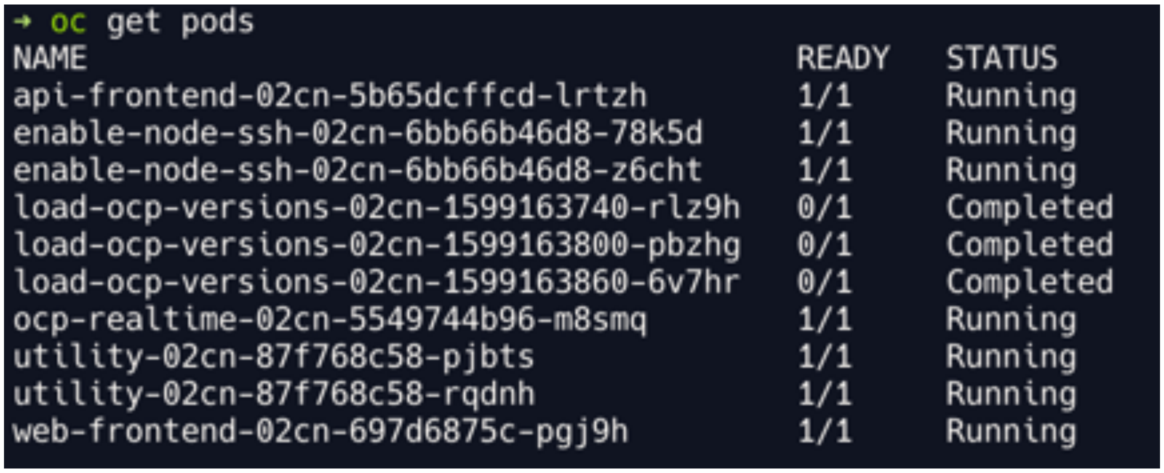
If you followed parts 9A and 9B, you created toolchains for the **frontend api** and **frontend web** microservices. Next, you will need to create additional pipelines for the following microservices:

- ocp-realtime-02cn
- enable-node-ssh-02cn
- load-ocp-versions-02cn
- utility-02cn

Choose your favorite method from with Classic or Tekton to create the remaining delivery pipelines.

After you have created all your delivery pipelines, make sure all of your pods are running. Log into your OpenShift cluster using the terminal and run:

```
oc get pods
```



```
→ oc get pods
NAME                                     READY   STATUS
api-frontend-02cn-5b65dcffcd-lrtzh       1/1     Running
enable-node-ssh-02cn-6bb66b46d8-78k5d    1/1     Running
enable-node-ssh-02cn-6bb66b46d8-z6cht    1/1     Running
load-ocp-versions-02cn-1599163740-rlz9h  0/1     Completed
load-ocp-versions-02cn-1599163800-pbzhg  0/1     Completed
load-ocp-versions-02cn-1599163860-6v7hr  0/1     Completed
ocp-realtime-02cn-5549744b96-m8smq       1/1     Running
utility-02cn-87f768c58-pjbts             1/1     Running
utility-02cn-87f768c58-rqdnh             1/1     Running
web-frontend-02cn-697d6875c-pgj9h        1/1     Running
```

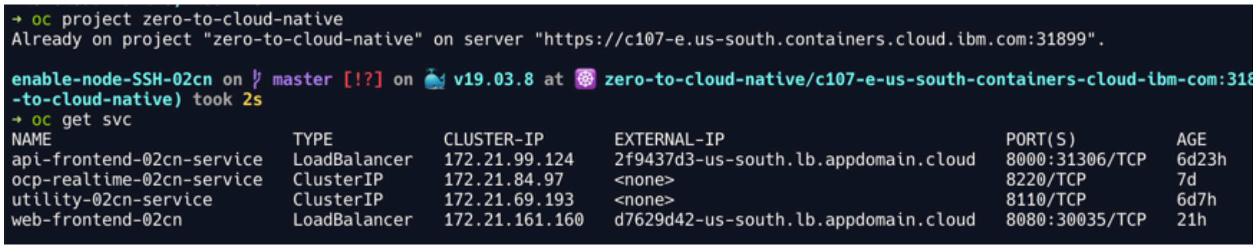
Your output should look like the above. Note that the load-ocp-versions-02cn microservice is a cronjob that runs every 5 minutes. You may see a different number of completions.

2 – Configure Cloud Internet Service

Part of the deployment.yaml files that were run from the frontend api and web delivery pipelines created a load balancer service. RedHat OpenShift on IBM Cloud will automatically create a VPC load balancer when you indicate in your deployment file that you want to create a load balancer service.

What we will need to do is map the load balancer url to a DNS entry for the domain we created with IBM Cloud Internet Services. The first thing we need to do is find the load balancer external IP.
To do so, log into your OpenShift cluster using the terminal. Once you are logged in, switch to the zero-to-cloud-native projects and list all the services using the following commands:

```
oc project zero-to-cloud-native
oc get svc
```



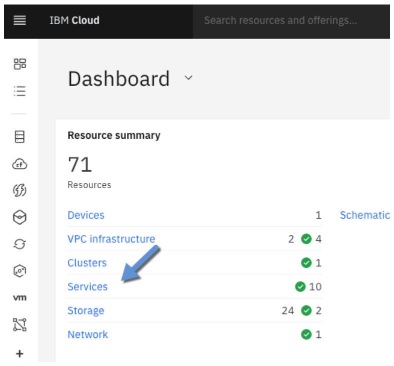
```
→ oc project zero-to-cloud-native
Already on project "zero-to-cloud-native" on server "https://c107-e.us-south.containers.cloud.ibm.com:31899".

enable-node-SSH-02cn on ᚠ master [!?] on 🐳 v19.03.8 at ☸ zero-to-cloud-native/c107-e-us-south-containers-cloud-ibm-com:318
-to-cloud-native) took 2s
→ oc get svc
NAME                        TYPE           CLUSTER-IP       EXTERNAL-IP                               PORT(S)          AGE
api-frontend-02cn-service   LoadBalancer   172.21.99.124    2f9437d3-us-south.lb.appdomain.cloud      8000:31306/TCP   6d23h
ocp-realtime-02cn-service   ClusterIP      172.21.84.97     <none>                                    8220/TCP         7d
utility-02cn-service        ClusterIP      172.21.69.193    <none>                                    8110/TCP         6d7h
web-frontend-02cn           LoadBalancer   172.21.161.160   d7629d42-us-south.lb.appdomain.cloud      8080:30035/TCP   21h
```

This will output 4 services.
api-frontend-02cn-service –external load balancer service for the api frontend.
ocp-realtime-02cn-service – this is an internal service for the ocp realtime microservice. This means the microservice is only available within the cluster. In our case, only the api-frontend will call this service.
utility-02cn-service – another internal service that provides common APIs that the other microservices will call.
web-frontend-02cn-service –external load balancer service for the web frontend.

Starting with the api frontend microservice, copy the external ip that you retrieved when you listed all of your services. In this case, **2f9437d3-us-south.lb.appdomain.cloud.** Note, you will have a different value.

Next, we need to create a DNS CNAME in Cloud Internet Services, mapping this external ip to the domain we created. From you IBM Cloud dashboard click on Services.


IBM Cloud Search resources and offerings...

Dashboard

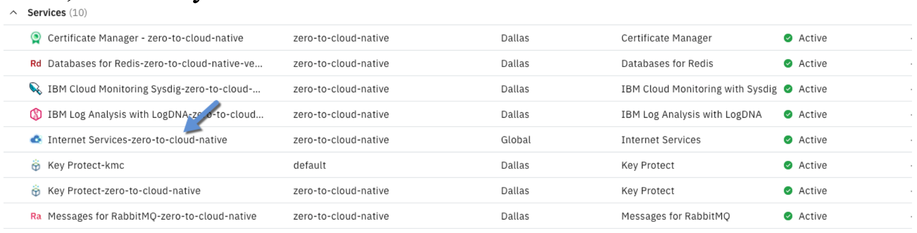
Resource summary

71
Resources

Devices	1	
VPC infrastructure	2	✓ 4
Clusters		✓ 1
Services		✓ 10
Storage	24	✓ 2
Network		✓ 1

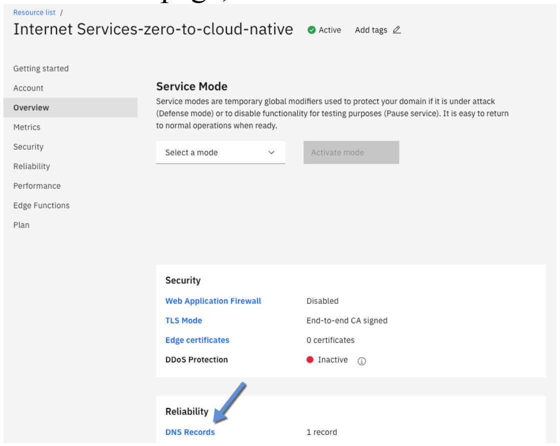
Schematic

Next, click on your Cloud Internet Services for the zero to cloud native tutorial.


Services (10)

Certificate Manager - zero-to-cloud-native	zero-to-cloud-native	Dallas	Certificate Manager	✓ Active
Databases for Redis-zero-to-cloud-native-ve...	zero-to-cloud-native	Dallas	Databases for Redis	✓ Active
IBM Cloud Monitoring Sysdig-zero-to-cloud-...	zero-to-cloud-native	Dallas	IBM Cloud Monitoring with Sysdig	✓ Active
IBM Log Analysis with LogDNA-zero-to-cloud...	zero-to-cloud-native	Dallas	IBM Log Analysis with LogDNA	✓ Active
Internet Services-zero-to-cloud-native	zero-to-cloud-native	Global	Internet Services	✓ Active
Key Protect-kmc	default	Dallas	Key Protect	✓ Active
Key Protect-zero-to-cloud-native	zero-to-cloud-native	Dallas	Key Protect	✓ Active
Messages for RabbitMQ-zero-to-cloud-native	zero-to-cloud-native	Dallas	Messages for RabbitMQ	✓ Active

On the next page, click on DNS Records.


Resource list /
Internet Services-zero-to-cloud-native ✓ Active Add tags

Service Mode
Service modes are temporary global modifiers used to protect your domain if it is under attack (Defense mode) or to disable functionality for testing purposes (Pause service). It is easy to return to normal operations when ready.

Security

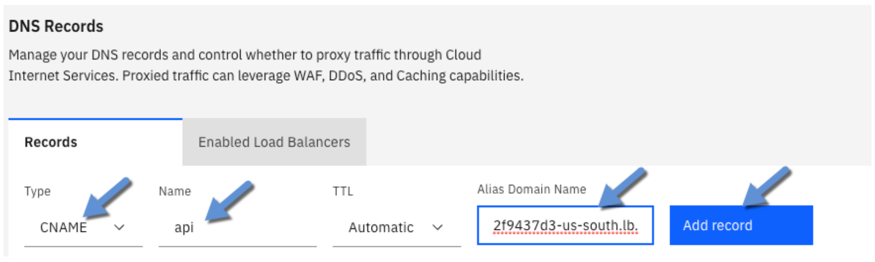
Web Application Firewall	Disabled
TLS Mode	End-to-end CA signed
Edge certificates	0 certificates
DDoS Protection	● Inactive

Reliability

DNS Records	1 record

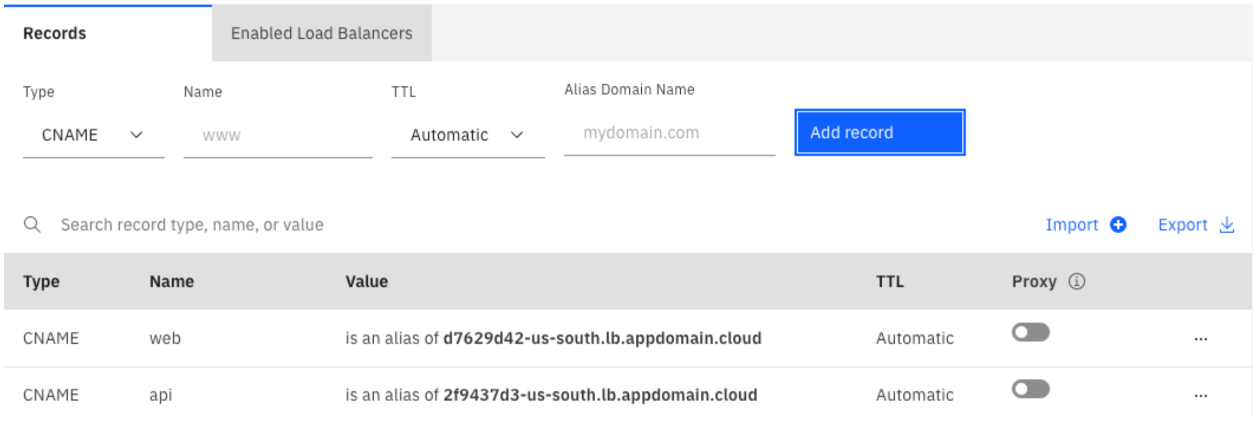
To create a DNS entry, we are going to create a CNAME DNS record.
Type: CNAME
Name: Past in the value you copied from the zero-to-cloud-native-api-service
Alias Doman: api.<the domain CIS is managing>

See an example below:


DNS Records
Manage your DNS records and control whether to proxy traffic through Cloud Internet Services. Proxied traffic can leverage WAF, DDoS, and Caching capabilities.

Type	Name	TTL	Alias Domain Name
CNAME	api	Automatic	2f9437d3-us-south.lb.

Repeat the same steps for web frontend service. After you are done, you should see two new DNS entries.



Type	Name	TTL	Alias Domain Name
CNAME	www	Automatic	mydomain.com

Search record type, name, or value

Type	Name	Value	TTL	Proxy
CNAME	web	is an alias of **d7629d42-us-south.lb.appdomain.cloud**	Automatic	
CNAME	api	is an alias of **2f9437d3-us-south.lb.appdomain.cloud**	Automatic	

Congratulations!! You have now successfully deployed and configured the zero to cloud native application. Now let’s test it!

3 – Test the application

As we have discussed, there are two frontend interfaces to the zero to cloud native tutorial application. A traditional web frontend and an API frontend.

3 – 1 Testing the API Application

To test the API interface, I will be using Postman which you installed in Part 5. Feel free to use any other API tool or even the curl command if you like.