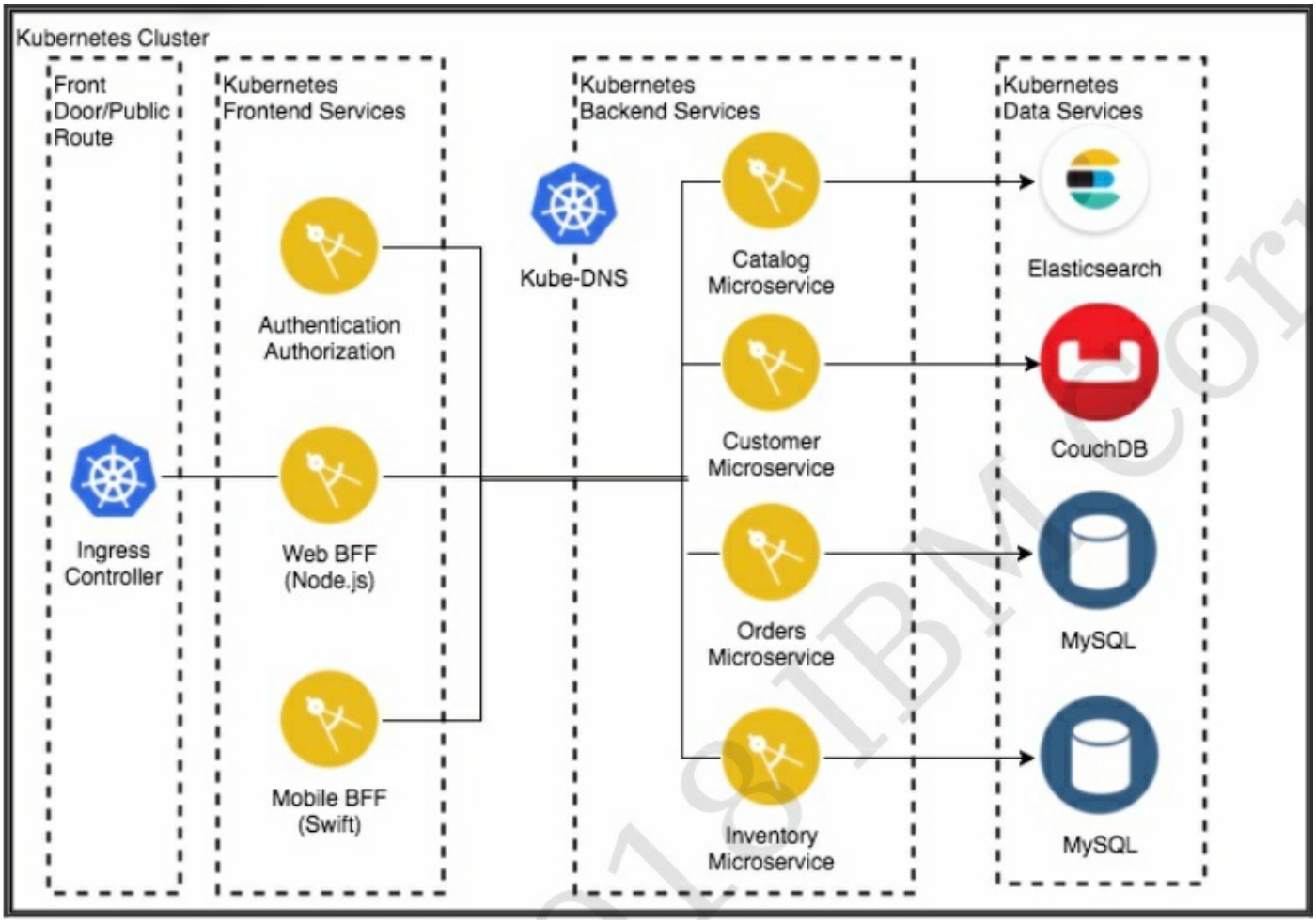
Deploying the BlueCompute Microservices Application

These exercises deploy the BlueCompute application. This microservices application is created as a showcase of developing IBM Cloud capabilities. You will use Jenkins to deploy the application components.

Exercise 1: Designing the Microservices Deployment

The application is shown in the following diagram.



1. The BlueCompute application contains a frontend layer, a backend layer (microservices) and a data layer (backing services).
2. Let's go through this exercise to design how you will group and deploy the application. The deployment should start with the data layer, followed by the backend layer and the frontend layer.
3. For the following data services resource, which Kubernetes resources should you define?
   - ElasticSearch: _____
   - CouchDB: _____
   - Orders MySQL: _____
   - Inventory MySQL: _____
4. For the following backend services microservice applications, which Kubernetes resources should you define?

- Catalog microservice: _____
- Customer microservice: _____
- Orders microservice: _____
- Inventory microservice: _____

5. For the following frontend services application, which Kubernetes resources should you define? Assume for now that the Ingress resource will be deployed with the Web application and that you will ignore the swift Web BFF.

   - Authentication: _____
   - Web BFF: _____

6. Now that you have been through the design part, let's consider the following:

   - What resource(s) is/are common for all components?
     _____
   - What resource(s) is/are common for the data layer?
     _____
   - When would you have an ingress resource?
     _____

Exercise 2: Setting Up and Verifying Backing Services

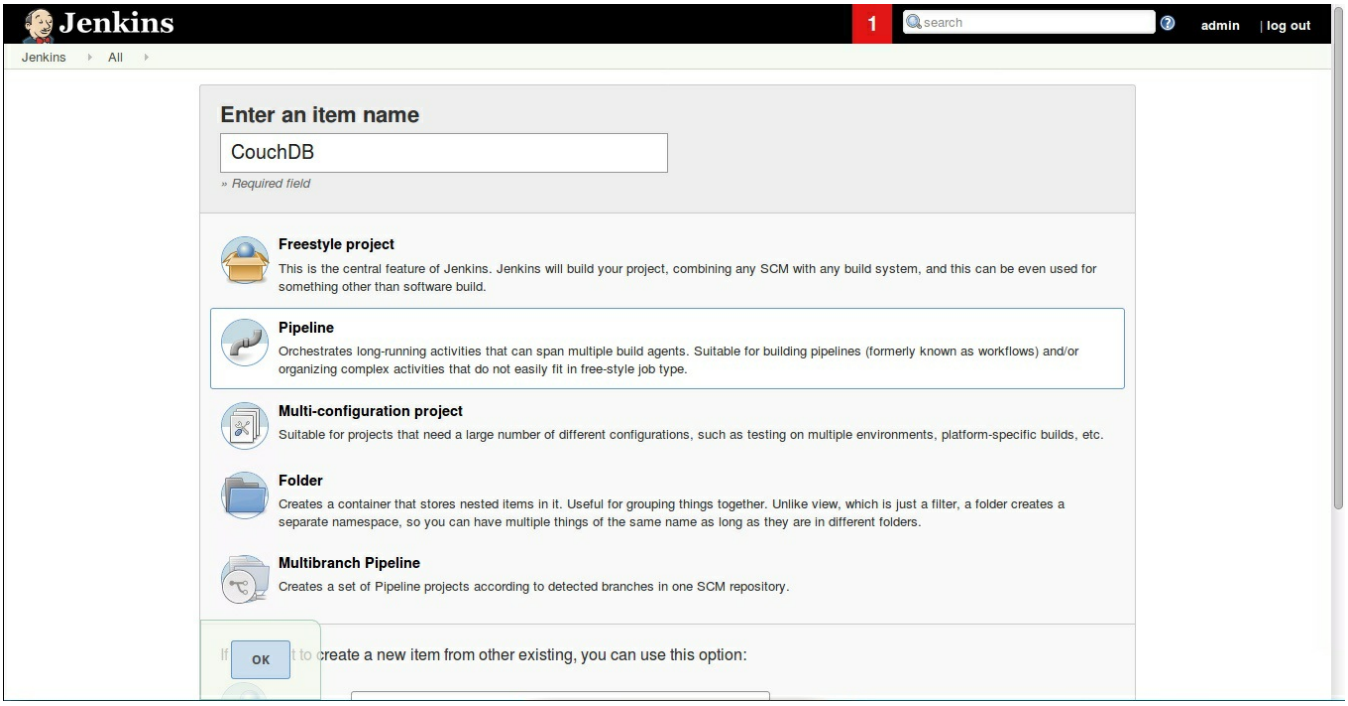
The following instructions allow you to build and deploy the backend services required by the BlueCompute application.

Note: Better than just following these steps, you can learn much more by looking at the Jenkinsfile and helm charts you will be working with. This will help you understand what actually happens in the process.

1. In your browser, go to the Jenkins Web UI at `http://10.10.1.4/jenkins`. Log in as `admin` with a password of `admin_0000`. You should be in the Jenkins dashboard.

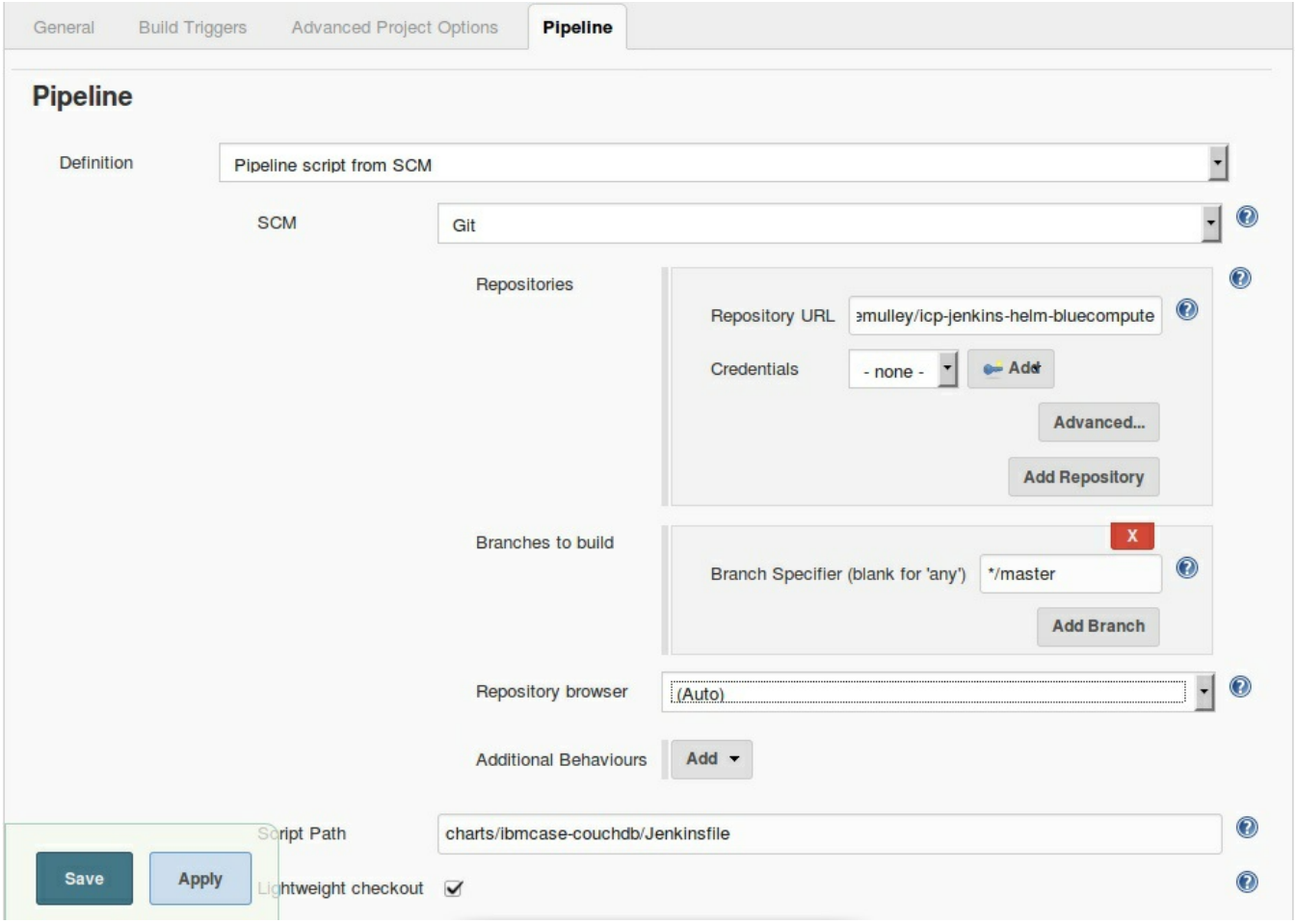
2. Click New item from the Jenkins menu on the left toolbar.

3. Enter a name of `CouchDB`, select Pipeline and click OK.



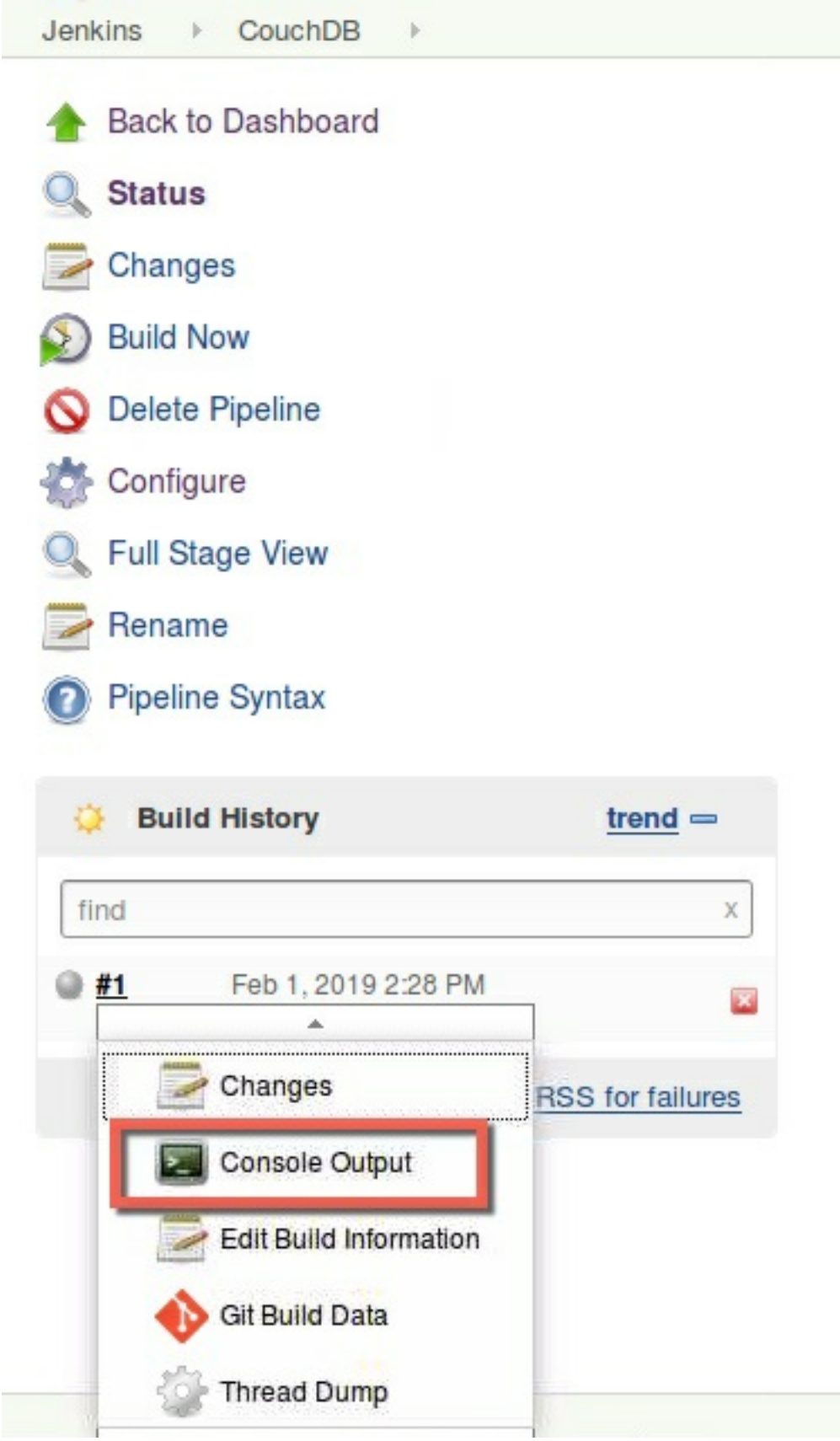
4. Scroll down to the Pipeline section and specify:

- Definition: `Pipeline script from SCM`
- SCM: `Git`
- Repository URL: `https://github.com/davemulley/icp-jenkins-helm-bluecompute`
- Branch Specifier: `*/master`
- Script Path: `charts/ibmcase-couchdb/Jenkinsfile`
- Click Save



5. On the couchDB pipeline page, click Build now.

6. Once the pipeline is running as indicated on the lower left side, open the drop down menu next to the run number and select Console Output.



7. The console should show the helm chart being deployed at the end and the pipeline having finished successfully.

```
==> v1beta1/Deployment
NAME                          DESIRED  CURRENT  UP-TO-DATE  AVAILABLE  AGE
bluecompute-couchdb--couchdb  1        1        1           0          0s

==> v1/Job
NAME                                                  DESIRED  SUCCESSFUL  AGE
bluecompute-couchdb-ibmcase-couchdb-create-user-x87ha  1        0           0s

==> v1/Pod(related)
NAME                                                     READY  STATUS             RESTARTS  AGE
bluecompute-couchdb--couchdb-7b4f857848-4dv5j            0/1    ContainerCreating  0         0s
bluecompute-couchdb-ibmcase-couchdb-create-user-x87ha-jk7f5  0/1    ContainerCreating  0         0s


NOTES:
1. Get the application URL by running these commands:
  export POD_NAME=$(kubectl get pods --namespace default -l "app=bluecompute-couchdb-ibmcase-couchdb" -o jsonpath="
{.items[0].metadata.name}")
  echo "Visit http://127.0.0.1:8080 to use your application"
  kubectl port-forward $POD_NAME 8080:5984

[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // container
[Pipeline] }
[Pipeline] // node
[Pipeline] }
[Pipeline] // podTemplate
[Pipeline] End of Pipeline
Finished: SUCCESS
```

8. Having the helm chart deployed does not necessarily mean that the application is correctly deployed. You must check whether the actual application pod is running. This may take a couple of minutes depending on the network speed to load the container. Run `kubectl get pod` commands and wait until the pod for couchdb is running.

```
root@master:~# kubectl get pods | grep couchdb
bluecompute-couchdb--couchdb-7b4f857848-4dv5j                1/1       Running     0          3m
bluecompute-couchdb-ibmcase-couchdb-create-user-x87ha-jk7f5  0/1       Completed   0          3m
root@master:~#
```

9. Check the status of the `bluecompute-couchdb` release, by running `helm status bluecompute-couchdb --tls`.

```
root@master:~# helm status bluecompute-couchdb --tls
LAST DEPLOYED: Fri Feb  1 06:29:05 2019
NAMESPACE: default
STATUS: DEPLOYED

RESOURCES:
==> v1/Secret
NAME                      TYPE    DATA  AGE
binding-customer-couchdb  Opaque  3     5m

==> v1/Service
NAME                          TYPE       CLUSTER-IP   EXTERNAL-IP  PORT(S)   AGE
bluecompute-couchdb-couchdb   ClusterIP  10.0.150.53  <none>       5984/TCP  5m

==> v1beta1/Deployment
NAME                          DESIRED  CURRENT  UP-TO-DATE  AVAILABLE  AGE
bluecompute-couchdb--couchdb  1        1        1           1          5m

==> v1/Job
NAME                                                  DESIRED  SUCCESSFUL  AGE
bluecompute-couchdb-ibmcase-couchdb-create-user-x87ha  1        1           5m

==> v1/Pod(related)
NAME                                                     READY  STATUS     RESTARTS  AGE
bluecompute-couchdb--couchdb-7b4f857848-4dv5j            1/1    Running    0         5m
bluecompute-couchdb-ibmcase-couchdb-create-user-x87ha-jk7f5  0/1    Completed  0         5m


NOTES:
1. Get the application URL by running these commands:
  export POD_NAME=$(kubectl get pods --namespace default -l "app=bluecompute-couchdb-ibmcase-couchdb" -o jsonpath="{.item
s[0].metadata.name}")
  echo "Visit http://127.0.0.1:8080 to use your application"
  kubectl port-forward $POD_NAME 8080:5984

root@master:~#
```

Can you identify the chart's components?

- Secret: `binding-customer-couchdb`

  _____

- Service: `bluecompute-couchdb-couchdb`

  _____

- Deployment: `bluecompute-couchdb--couchdb`

  _____

- Job: `bluecompute-couchdb-ibmcase-couchdb-create-user-"*"`

  _____

- Pods: `bluecompute-couchdb--couchdb-"*"` and `bluecompute-couchdb-ibmcase-couchdb-create-user-"*"`

  _____

Note: The deployment of the other backend components (elasticsearch, inventory-mysql and orders-mysql) is actually very similar. You create the Jenkins pipelines from SCM (GIT) and run them individually. These pipelines are not automated (triggered using code changes) as this is meant to be a stable backend. In a real production environment, you would want to add a PersistentVolumeClaim to physically host the data instead of storing it in volatile containers as this example describes.

10. Deploy `elasticsearch` by creating a new pipeline, with the following parameters:

    - Name: `Elasticsearch`
    - Repository URL: `https://github.com/davemulley/icp-jenkins-helm-bluecompute`
    - Branch Specifier: `*/master`
    - Script Path: `charts/ibmcase-elasticsearch/Jenkinsfile`
    - Run the pipeline using the Build now link.
    - Make sure the `bluecompute-elasticsearch` pod is running.

11. Check the status of the `bluecompute-elasticsearch` release, by running `helm status bluecompute-elasticsearch --tls`.

```
root@master:~# helm status bluecompute-elasticsearch --tls
LAST DEPLOYED: Fri Feb  1 06:40:07 2019
NAMESPACE: default
STATUS: DEPLOYED

RESOURCES:
==> v1/Secret
NAME                          TYPE    DATA  AGE
binding-catalog-elasticsearch  Opaque  1     54s

==> v1/Service
NAME                                          TYPE       CLUSTER-IP   EXTERNAL-IP  PORT(S)            AGE
bluecompute-elasticsearch-catalog-elasticsearch  ClusterIP  10.0.37.236  <none>       9200/TCP,9300/TCP  54s

==> v1beta1/Deployment
NAME                                            DESIRED  CURRENT  UP-TO-DATE  AVAILABLE  AGE
bluecompute-elasticsearch-catalogdb-elasticsearch  1        1        1           1          54s

==> v1/Pod(related)
NAME                                                      READY  STATUS   RESTARTS  AGE
bluecompute-elasticsearch-catalogdb-elasticsearch-5b85cdbccjgd6  1/1    Running  0         54s


root@master:~#
```

Can you identify the chart's components?

- Secret: `binding-catalog-elasticsearch`

  _____

- Service: `bluecompute-elasticsearch-catalog-elasticsearch`

  _____

- Deployment: `bluecompute-elasticsearch-catalogdb-elasticsearch`

  _____

- Pod: `bluecompute-elasticsearch-catalogdb-elasticsearch` -

  _____

12. Deploy `inventory-mysql` by creating a new pipeline, with the following parameters:

    - Name: `Inventory-mysql`
    - Repository URL: `https://github.com/davemulley/icp-jenkins-helm-bluecompute`
    - Branch Specifier: `*/master`
    - Script Path: `charts/ibmcase-inventory-mysql/Jenkinsfile`
    - Run the pipeline using the Build now link.
    - Make sure the `bluecompute-inventory-mysql` pod is running

13. Check the status of the `bluecompute-inventory-mysql` release, by running `helm status bluecompute-inventory-mysql --tls`.

```
root@master:~# helm status bluecompute-inventory-mysql --tls
LAST DEPLOYED: Fri Feb  1 06:43:26 2019
NAMESPACE: default
STATUS: DEPLOYED

RESOURCES:
==> v1/Job
NAME                                                          DESIRED  SUCCESSFUL  AGE
bluecompute-inventory-mysql-ibmcase-inventory-mysql-populate-my  1        0           1m

==> v1/Pod(related)
NAME                                                          READY  STATUS   RESTARTS  AGE
bluecompute-inventory-mysql-inventory-mysql-5d98d64589-kbl5l     1/1    Running  0         1m
bluecompute-inventory-mysql-ibmcase-inventory-mysql-populasxjg8  1/1    Running  0         1m

==> v1/Secret
NAME                       TYPE    DATA  AGE
binding-inventorydb-mysql  Opaque  4     1m

==> v1/Service
NAME                                        TYPE       CLUSTER-IP    EXTERNAL-IP  PORT(S)   AGE
bluecompute-inventory-mysql-inventory-mysql  ClusterIP  10.0.225.192  <none>       3306/TCP  1m

==> v1beta1/Deployment
NAME                                        DESIRED  CURRENT  UP-TO-DATE  AVAILABLE  AGE
bluecompute-inventory-mysql-inventory-mysql  1        1        1           1          1m


root@master:~#
```

    Can you identify the chart's components?

    - Secret: `binding-inventorydb-mysql`

      _____

    - Service: `bluecompute-inventory-mysql-inventory-mysql`

      _____

    - Deployment: `bluecompute-inventory-mysql-inventory-mysql`

      _____

    - Job: `bluecompute-inventory-mysql-ibmcase-inventory-mysql-populate*`

      _____

    - Pods: `bluecompute-inventory-mysql-inventory-mysql-*` and `bluecompute-inventory-mysql-ibmcase-inventory-mysql-popu*`

      _____

14. Deploy `orders-mysql` by creating a new pipeline, with the following parameters:
    - Name: `Orders-mysql`
    - Repository URL: `https://github.com/davemulley/icp-jenkins-helm-bluecompute`
    - Branch Specifier: `*/master`
    - Script Path: `charts/ibmcase-orders-mysql/Jenkinsfile`
    - Run the pipeline using the Build now link.
    - Make sure the `bluecompute-orders-mysql` pod is running
15. Check the status of the `bluecompute-orders-mysql` release, by running `helm status bluecompute-orders-mysql --tls`.

```
root@master:~# helm status bluecompute-orders-mysql --tls
LAST DEPLOYED: Fri Feb  1 06:47:49 2019
NAMESPACE: default
STATUS: DEPLOYED

RESOURCES:
==> v1/Secret
NAME                    TYPE    DATA  AGE
binding-ordersdb-mysql  Opaque  4     43s

==> v1/Service
NAME                                  TYPE       CLUSTER-IP   EXTERNAL-IP  PORT(S)   AGE
bluecompute-orders-mysql-orders-mysql  ClusterIP  10.0.18.244  <none>       3306/TCP  43s

==> v1beta1/Deployment
NAME                                  DESIRED  CURRENT  UP-TO-DATE  AVAILABLE  AGE
bluecompute-orders-mysql-orders-mysql  1        1        1           1          43s

==> v1/Pod(related)
NAME                                                   READY  STATUS   RESTARTS  AGE
bluecompute-orders-mysql-orders-mysql-86d7449777-5sh64  1/1    Running  0         43s


root@master:~#
```

    Can you identify the chart's components?

    - Secret: `binding-ordersdb-mysql`

      _____
    - Service: `bluecompute-orders-mysql-orders-mysql`

      _____
    - Deployment: `bluecompute-orders-mysql-orders-mysql`

      _____
    - Pod: `bluecompute-orders-mysql-orders-mysql-*`

      _____
16. Answer the following questions:
    - How would you get the URL and credentials to access the backend services?

      _____

      _____
    - Why do some charts contain job(s) while other don't?

      _____
    - What other resource is typically present for a backend service?

      _____

Exercise 3: Deploying Microservices Components

The set of charts in this exercise is different that in the previous set. The main difference is that the docker images are built by the pipeline and loaded to the IBM Cloud Private local registry.

Note:

These instructions load the Jenkinsfile directly from a forked repository that has been pre-configured. The actual steps should have been:

- Fork the repository.
- Create a Jenkinsfile.
- Poll the repository for commit changes and create a post-commit hook.

You would never need to build the pipeline manually.

1. There are 4 microservices built into the BlueCompute application: customer, inventory, catalog and orders. You will deploy them similar to deploying the backend resources, but using different source git repositories.

2. Create a new Jenkins pipeline using the following specifications for the `customer` application.

    - Name: `Customer`
    - Repository URL: `https://github.com/davemulley/refarch-cloudnative-micro-customer`
    - Branch Specifier: `*/master`
    - Script Path: `Jenkinsfile`
    - Run the pipeline using the Build now link.
    - Make sure the `customer` pod is running

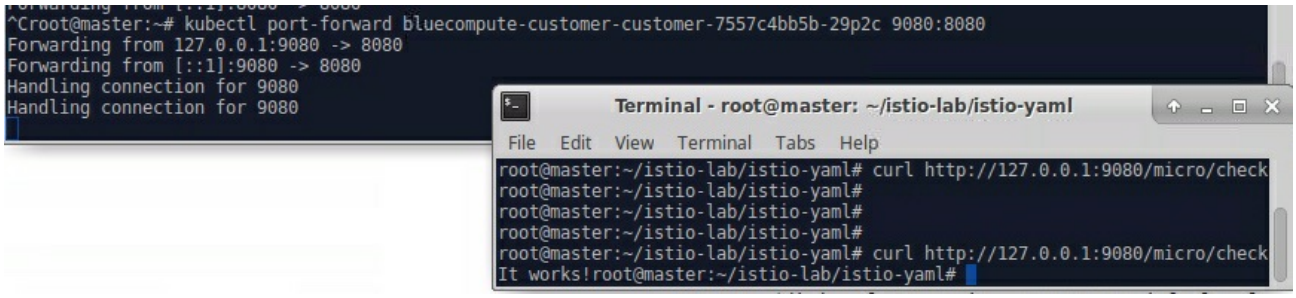
3. Check the status of the `bluecompute-customer` release, by running `helm status bluecompute-customer --tls`.

```
root@master:~# helm status bluecompute-customer --tls
LAST DEPLOYED: Fri Feb  1 08:03:22 2019
NAMESPACE: default
STATUS: DEPLOYED

RESOURCES:
==> v1/Pod(related)
NAME                                                    READY  STATUS     RESTARTS  AGE
bluecompute-customer-customer-7557c4bb5b-29p2c          1/1    Running    0         1m
bluecompute-customer-customer-create-user-zqkap-677nl   0/1    Error      0         1m
bluecompute-customer-customer-create-user-zqkap-dq2zz   0/1    Error      0         1m
bluecompute-customer-customer-create-user-zqkap-xrvp4   0/1    Completed  0         1m

==> v1/Secret
NAME                                     TYPE    DATA  AGE
bluecompute-customer-customer-hs256-key  Opaque  1     1m

==> v1/Service
NAME                           TYPE       CLUSTER-IP    EXTERNAL-IP  PORT(S)   AGE
bluecompute-customer-customer  ClusterIP  10.0.112.131  <none>       8080/TCP  1m

==> v1beta1/Deployment
NAME                           DESIRED  CURRENT  UP-TO-DATE  AVAILABLE  AGE
bluecompute-customer-customer  1        1        1           1          1m

==> v1/Job
NAME                                           DESIRED  SUCCESSFUL  AGE
bluecompute-customer-customer-create-user-zqkap  1        1           1m


NOTES:
1. Get the application URL by running these commands:
  export POD_NAME=$(kubectl get pods --namespace default -l "app=bluecompute-customer-customer" -o jsonpath="{.items[0].m
etadata.name}")
  echo "Visit http://127.0.0.1:8080 to use your application"
  kubectl port-forward $POD_NAME 8080:8080

root@master:~#
```

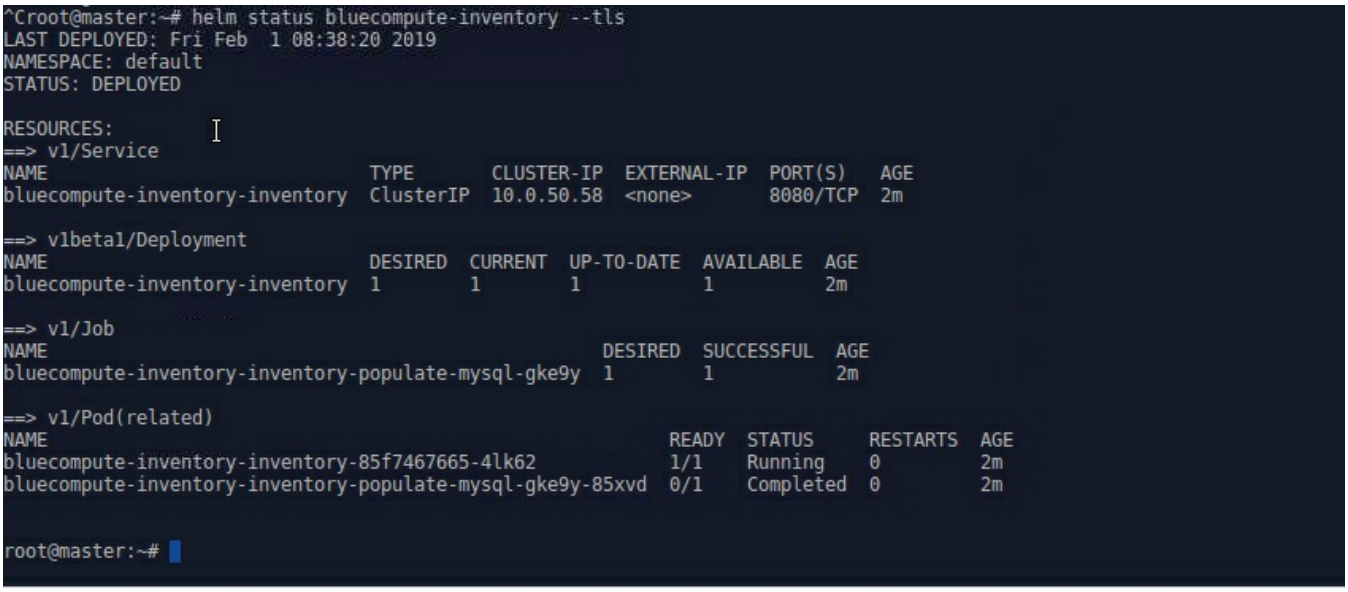
4. Test the `customer` application's health:
   - Open a terminal window
   - Run the command `kubectl port-forward <customerpod> 9080:8080` The command maps the pod 8080 to the localhost.
   - Open another terminal window and check the microservice. Note that most of the usual microservice REST calls for the `customer` application are protected by JWT. Therefore, you can only easily test the check call which returns the string `It works`.

     ```
     curl http://127.0.0.1:9080/micro/check
     ```

   - Stop the port forwarding by typing `Ctrl-C`.



5. Create a new Jenkins pipeline using the following specifications for the `inventory` application.
   - Name: `Inventory`
   - Repository URL: `https://github.com/davemulley/refarch-cloudnative-micro-inventory`
   - Branch Specifier: `*/master`
   - Script Path: `inventory/Jenkinsfile`
   - Run the pipeline using the Build now link.
   - Make sure the `inventory` pod is running
6. Check the status of the `bluecompute-inventory` release, by running `helm status bluecompute-inventory --tls`.



7. Test the `inventory` application's health:
   - Open a terminal window
   - Run the command `kubectl port-forward <inventorypod> 9080:8080` The command maps the pod 8080 to the localhost port 8080.
   - Open another terminal window and check the microservices. Note that this microservice is not secured (how do you know that? _____)

     ```
     curl http://127.0.0.1:9080/micro/inventory
     ```

   - Stop the port forwarding by typing `Ctrl-C`.

8. Create a new Jenkins pipeline using the following specifications for the `catalog` application. Note that the repository below is correct. The `catalog` application is in the `inventory` repository.
   - Name: `Catalog`
   - Repository URL: `https://github.com/ibm-cloud-academy/refarch-cloudnative-micro-inventory`
   - Branch Specifier: `*/master`
   - Script Path: `catalog/Jenkinsfile`
   - Run the pipeline using the Build now link.
   - Make sure the `catalog` pod is running
9. Check the status of the `bluecompute-catalog` release, by running `helm status --tls bluecompute-catalog`.

10. Test the `catalog` application's health:
    - Open a terminal window
    - Run the command `kubectl port-forward <catalogpod> 8081:8081` The command maps the pod 8081 to the localhost port 8081.
    - How do you know that this application uses port 8081? _____
    - Open another terminal window and check the microservice. Note that this microservice is not secured

      ```
      curl http://127.0.0.1:8081/micro/items/13401
      ```

    - Stop the port forwarding by typing `Ctrl-C`.

11. Create a new Jenkins pipeline using the following specifications for the `orders` application.

- Name: `Orders`
- Repository URL: `https://github.com/ibm-cloud-academy/refarch-cloudnative-micro-orders`
- Branch Specifier: `*/master`
- Script Path: `Jenkinsfile`
- Run the pipeline using the Build now link.
- Make sure the `orders` pod is running

12. Check the status of the `bluecompute-orders` release, by running `helm status --tls bluecompute-orders` .

    ❓

13. Test the `orders` application's health:
    - Open a terminal window
    - Run the command `kubectl port-forward <orderspod> 8080:8080` The command maps the pod 8080 to the localhost port 8080.
    - Is this microservice being secured using JWT? ____

      ```
      curl http://127.0.0.1:8080/micro/check
      ```

    - Stop the port forwarding by typing `Ctrl-C` .

      ❓

14. Create a new Jenkins pipeline using the following specifications for the `authentication` application.
    - Name: `Auth`
    - Repository URL: `https://github.com/ibm-cloud-academy/refarch-cloudnative-auth`
    - Branch Specifier: `*/master`
    - Script Path: `Jenkinsfile`
    - Run the pipeline using the Build now link.
    - Make sure the `auth` pod is running
15. Check the status of the `bluecompute-auth` release, by running `helm status --tls bluecompute-auth` .

    ❓

16. You will test the oauth authentication later after the Web application has been deployed.
17. Create a new Jenkins pipeline using the following specifications for the `Web` application.
    - Name: `Web`
    - Repository URL: `https://github.com/ibm-cloud-academy/refarch-cloudnative-bluecompute-web`

- Branch Specifier: `*/master`
- Script Path: `Jenkinsfile`
- Run the pipeline using the Build now link.
- Make sure the `bluecompute-web` pod is running

18. Check the status of the `bluecompute-web` release, by running `helm status --tls bluecompute-web`.

    ❓

19. What resources are created in the Web application? What are the usage of those resources?

    - ConfigMap:

      _____

      _____

    - Ingress:

      _____

      _____

20. The final state of the Jenkins dashboard should be similar to the following:

    ❓

Exercise 4: Validating the Application and Resources

Now that you have completed all the deployments successfully, it's time to test the overall microservice application.

1. Open the URL to get to the BlueCompute web application: `http://proxy/bluecompute`

   ❓

2. Click the `Log in` link and log in with as user `foo` and a password of `bar`.

   ❓

3. After login, the Catalog view is shown. This verifies that the `catalog` microservice is working.

   ❓

4. Select an item. The example below shows the Dayton Meat Chopper.

   ❓

5. From the Place your order here dropdown, select a number and click Buy. This checks whether the `Orders` microservice is working. You should be able to retrieve your orders in the Profile page.

6. Click the Profile link. Verify that it can retrieve your user profile and the orders you have placed.

   ❓

*** End of exercises ***