VMware Tanzu

Planning and Architecture

YugabyteDB on Tanzu Reference Architecture

Michael James, Solutions Architect

May 5, 2022

YugabyteDB on Tanzu Reference Architecture

Planning and Operations

Architecture Overview

This document details a reference architecture for deploying YugabyteDB on Tanzu Kubernetes Grid (TKG). This reference will cover topics such as Kubernetes requirements and cluster layout for YugabyteDB.

This architecture should give you a path to creating a highly available, production-grade deployment of YugabyteDB. However you should not feel constrained by this exact path if your specific use cases lead you to a different deployment architecture. Design decisions in this architecture reflect the main design issues and the rationale behind a chosen solution path and if necessary can help provide rationale for any deviation.

Cluster layout

The level of availability and redundancy required by the workloads being deployed will determine the topology of the clusters, from a simpler, single cluster running in just one availability zone, to a more complex deployment of multiple clusters distributed all over the world, either on a single cloud provider, or across multiple cloud providers.

Bill of Materials

Component	Version
Tanzu Kubernetes Grid	1.5.1
Kubernetes	1.22.5
YugabyteDB Anywhere	2.13.0
YCQL CLI	2.11.2.0

This reference architecture was validated using these versions of Tanzu and Yugabyte solutions that are in scope.

Design Use-Cases

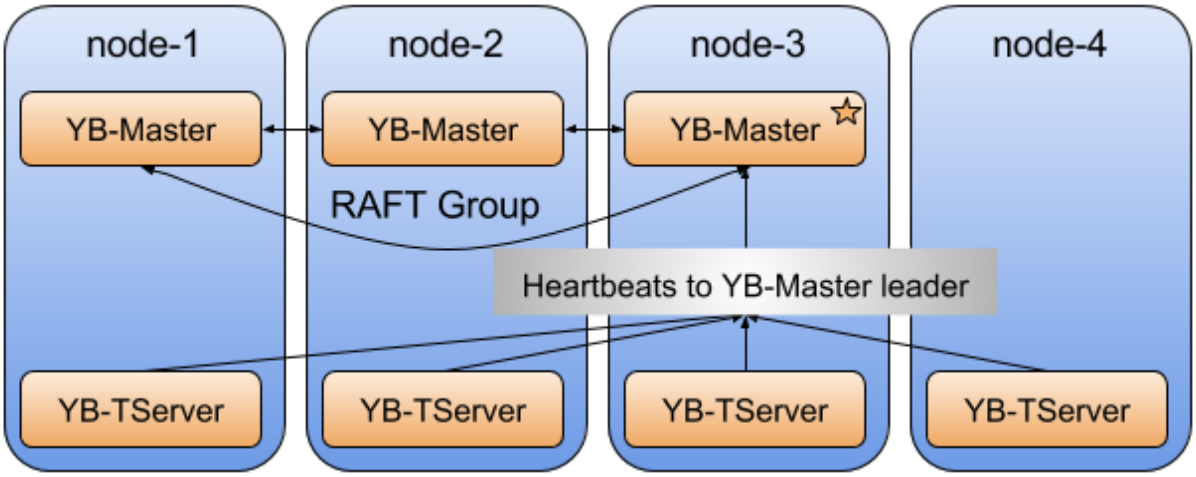
Design #	Decision	Rationalization	Ramifications
TKG-001	A single cluster in one region and one availability zone.	Appropriate choice for a development/test, non-critical environment where availability, or loss of data is of little to no concern. Primary benefit of this design is ease of deployment.	Not an option for workloads requiring high availability, durability, and resiliency.
TKG-002	A single cluster in one region spread across multiple availability zones.	Development/test or production workloads that need to scale, and require a level of availability and durability in the event of a localized failure to a single data center, but does not need to be distributed across a wider region.	A regional failure makes this option unacceptable for broader, distributed workloads.
TKG-003*	Multiple clusters spread across multiple regions across multiple availability zones.	Production workloads that demand the highest level of availability and resiliency spanning geo-locations nationally and/or internationally.	Maintaining a level of consistency across all deployments becomes paramount.
TKG-004*	Multiple clusters spread across multiple cloud providers.	Production workloads that demand the highest level of resiliency, but unlike TKG-003, must be portable across multiple public cloud providers and private on-premises environments.	Highest level of complexity, keeping data consistent.

**Use-cases TKG-003 and TKG-004 are currently not supported with Kubernetes without federation across regions. Tanzu Service Mesh addresses this requirement by grouping all Kubernetes clusters across regions and/or cloud providers as a single entity under that which is called a Global Namespace.*

YugabyteDB Components

The following components are installed and run on one or more Kubernetes clusters.

- **YugabyteDB Anywhere**, the management plane for hosting the console for creating and managing YugabyteDB Universes. YugabyteDB Anywhere was formerly referred to as Yugabyte Platform.
- **YugabyteDB Universe(s)**, collection of nodes, or cluster(s), that host the YB-Master and YB-TServer components. It is the environment that defines the resources necessary to host a sharded database.
- **YB-Master**, maintains system metadata, location and number of tables and access controls, DDL, replication, and load balancing operations.
- **YB-TServer**, handles input/output operations to data partitioned and replicated across the cluster. APIs are exposed via standard ports applicable to the underlying database technology, i.e., PostgresQL (5433) and Cassandra (9042).



High-level overview of a YugabyteDB Universe.

https://docs.yugabyte.com/latest/architecture/

Cluster Requirements

General Infrastructure Recommendations

Storage

- Local SSD (recommended), lowest latency, highest performance.
- Two disks per node.
- When storage requirements increase, the recommendation is to add more nodes, not increase size of disks.
- vSAN Flash

> “Mount settings - XFS is the recommended filesystem. - Use the noatime setting when mounting the data drives. - ZFS isn’t currently supported and is on the roadmap. - NFS isn’t currently supported and is on the roadmap.”
>
> “So XFS can come from VMDK on any VMFS datastore really”
>
> 6000 IOPS
>
> SAN mounted, vSAN

Compute

> Use the following CPU/RAM/DISK ratio as a general guideline.
>
> *8 vCPU / 16 GB RAM / 400 GB storage*
>
> As shown in the diagram above, the number of nodes should be equal to or greater than the number of yb-master pods so they can be distributed evenly. The number of yb-masters equates to the replication factor of the database. The replication factor will be the number of redundant copies, or replicas, of the database.

For more information, refer to the deployment checklist here.

https://docs.yugabyte.com/preview/deploy/checklist/

Single AZ - Single Cluster

The following components and their respective specifications are required for hosting a minimal installation of YugabyteDB Anywhere and one YugabyteDB Universe.

It is generally recommended for production environments to install YugabyteDB Anywhere on its own dedicated cluster. However, for development purposes, the suggestion made here is to host on the same cluster, in its own namespace.

1 TKG Management Cluster

- 1 control plane node
- 1 worker node

1 Tanzu Kubernetes Cluster

- 1 control plane node
- 4* worker nodes (*1 node = 8 CPU, 32GB RAM, 80GB Storage)*

  ***The size of the nodes will determine the number of nodes needed, or vice versa. The larger the node size, the number of nodes can be reduced.*

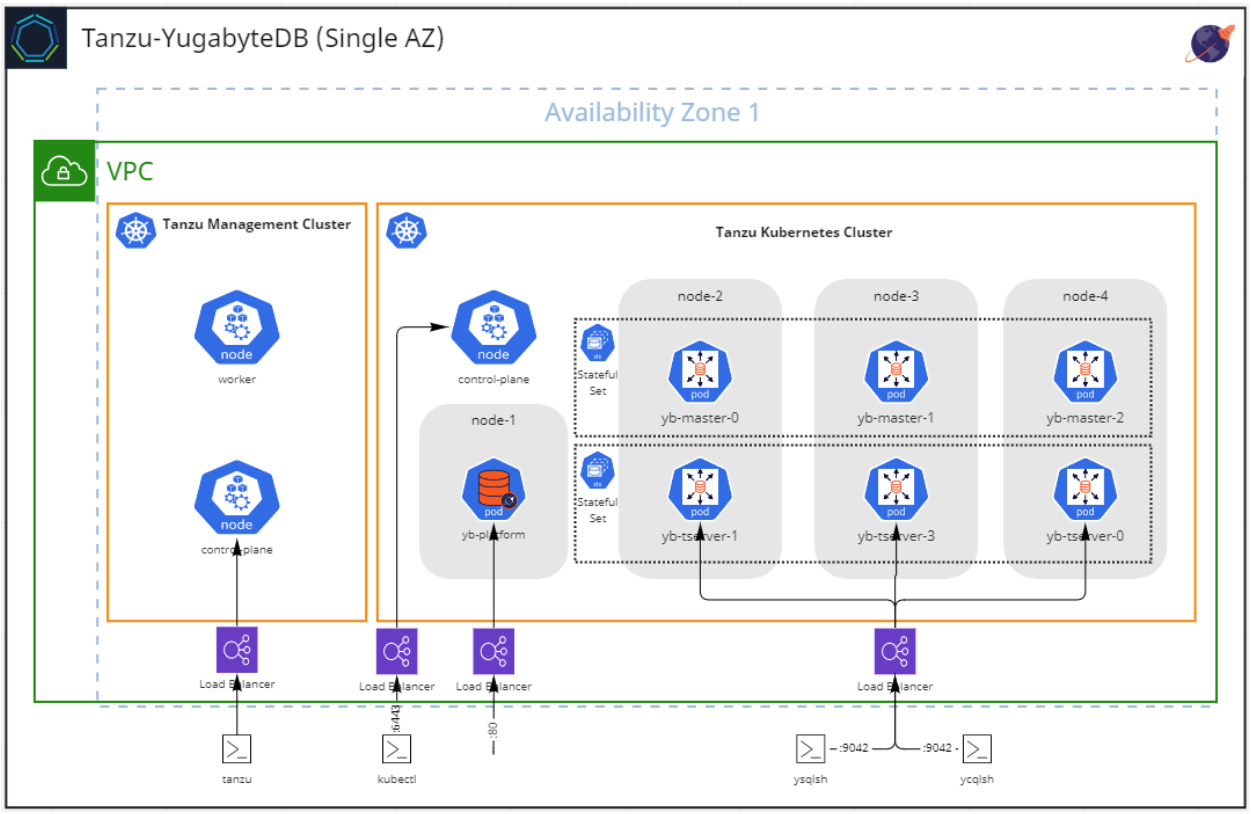
YugabyteDB

- YugabyteDB Anywhere
- At least 1 Universe

Kubernetes requirements:

1. Load Balancer to YB-TServer
2. Default storage class, default TKG 80GB is good starting point
3. At least 8GB per node

Architectural View



NOTE: The yb-platform can be run in the management cluster.

Single Region - Multiple AZs

The following components and their respective specifications are required for hosting a moderately available installation of YugabyteDB Anywhere and at least one or more YugabyteDB Universes.

It is generally recommended for production environments to install YugabyteDB Anywhere on its own dedicated cluster, which will be followed here. No more than one AZ is typically needed.

Tanzu Management Cluster

- 3 control plane nodes
- 3 worker nodes

Tanzu Kubernetes Cluster

- 3 control plane nodes
- 4* worker nodes (*1 node = 8 CPU, 32GB RAM, 80GB Storage)*

  ***The size of the nodes will determine the number of nodes needed, or vice versa. The larger the node size, the number of nodes can be reduced.*

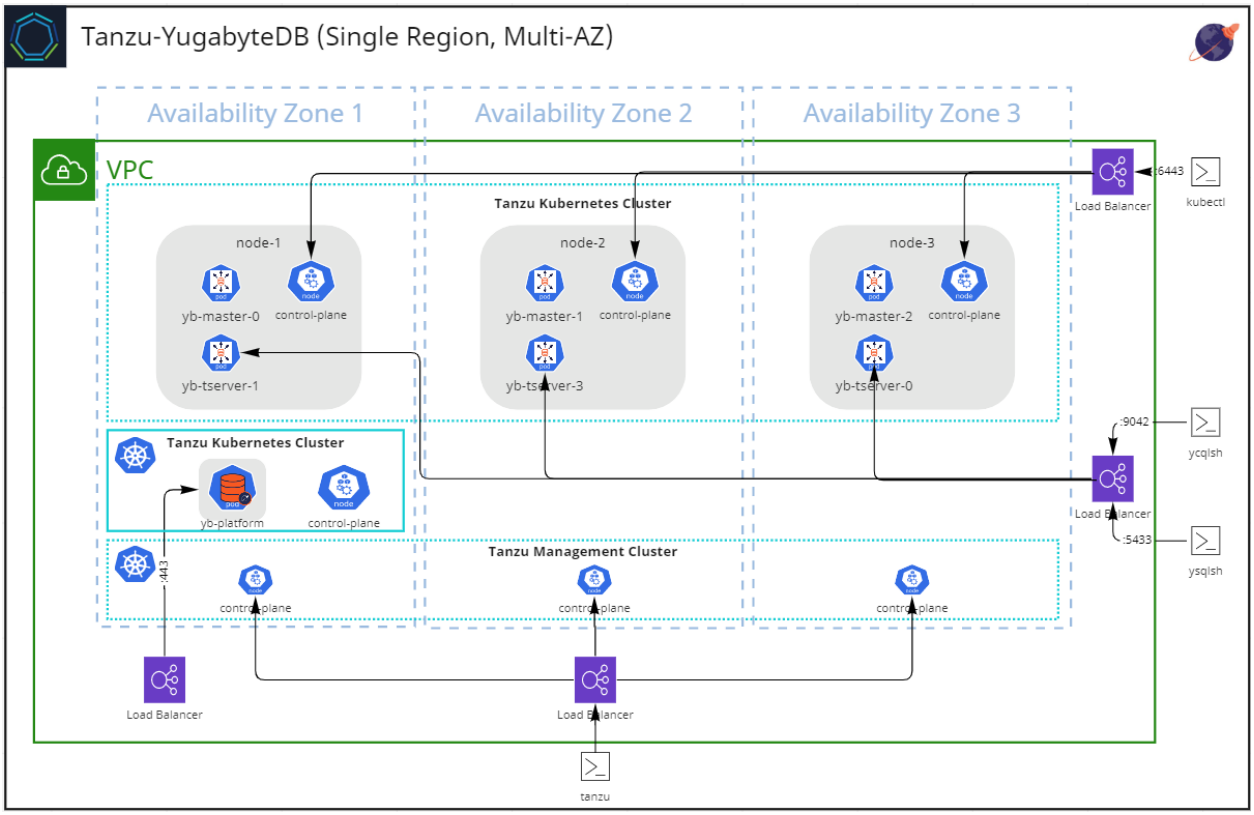
YugabyteDB

- YugabyteDB Anywhere
- At least 1 Universe

Kubernetes requirements:

4. Load Balancer to YB-TServer
5. Default storage class (80GB is good starting point)
6. At least 8GB per node

Architectural View



Backup and Recovery

Each underlying database has its own mechanism for backing up data. The following sections outline options for simple export of data to an automated generation of snapshots of the data.

KUBERNETES

The recommended backup solution for YugabyteDB Anywhere itself is Velero. This will enable backups of metadata of Universes admins create and manage.

EXPORT

Use this option to import the data into another database solution than the source.

Cassandra

For backups, use the *ycqlsh* CLI in the bin directory of the YugabyteDB package with either of the following two options.

- *DESCRIBE (DESC)*, copies the schema for either a single keyspace or all keyspaces within the database to an output file.
- *COPY TO*, copies the actual data rows to an output file.

To restore, use the same ycqlsh CLI with either of the following options.

- *SOURCE*, for example, ycqlsh -e “SOURCE ‘my-describe-backup.cql’”
- *COPY FROM*, loads data into a keyspace schema that was output as a file using the COPY TO command.

PostgresQL

Two options are available in the postgres/bin directory of the YugabyteDB package..

- *ysql_dump*, backs up a single database into an SQL script file. This file will also be used for the restoration of the database.
- *ysql_dumpall*, backs up all databases in a single universe, including all of the associated metadata.

Use the *ysqlsh* CLI with the output file from one of the above commands to restore the database.

For more information, refer to the YugabyteDB documentation.

https://docs.yugabyte.com/preview/manage/backup-restore/export-import-data

SNAPSHOT

Snapshots of the data can be generated on-demand or on an automated schedule with the yb-admin CLI. These snapshots are stored alongside the data in the same cluster by default, in which case, the ability to restore to a specific point in time, or can be stored in dedicated storage, such as NFS, or public storage service, such as AWS S3 or Azure Blob Storage.

For more information, refer to the YugabyteDB documentation.

https://docs.yugabyte.com/preview/manage/backup-restore/snapshot-ysql

YugabyteDB Anywhere provides API and UI for seamless integration with external storage.

https://docs.yugabyte.com/preview/yugabyte-platform/back-up-restore-universes

POINT-IN-TIME RECOVERY

Point-in-time recovery (PITR) minimizes the RPO by restoring to the latest known working state of a database, as opposed to a time of snapshot creation.

PITR is based on snapshots that are automatically created and stored in the cluster alongside the data.

For more information, refer to the YugabyteDB documentation.

https://docs.yugabyte.com/preview/manage/backup-restore/point-in-time-recovery

Monitoring

The following are options for monitoring real-time universe metrics and statistics.

- **YugabyteDB Anywhere**, includes embedded charts and graphs showing latency, number of I/O operations, etc...

- **Tanzu Observability**, an advanced service for collecting metrics, traces, and logs. Includes 200+ integrations with the most popular services and libraries, including Prometheus, PostgresQL, and Cassandra.
- **Prometheus**, component for collecting host metrics and outputting to charting components.

Logging

The yugabyte-data storage volume contains subdirectories per each node of a cluster. Each component will have a log directory specific to its function.

- **yb-master**, yugabyte-data/disk1/yb-data/master/logs/
- **yb-tserver**, yugabyte-data/disk1/yb-data/tserver/logs/

Deployment

Tanzu-Yugabyte

Recommended Prerequisites (Multi-AZ)

Component	Version
Tanzu Kubernetes Grid	1.5.1
Kubernetes	1.22.5
Helm	3.4 or later
YugabyteDB (trial required)	2.13.0
YSQLSH CLI YCQLSH CLI	2.11.2.0
Python	2.7 or later

Instructions and installation scripts can be downloaded from the following repository.

https://github.com/nycpivot/tanzu-yugabyte

Details can be found at the following YugabyteDB docs.

https://docs.yugabyte.com/latest/

https://docs.yugabyte.com/latest/quick-start/install/kubernetes/

Overview

The installation process basically consists of 3 stages (assuming pre-existing cluster)..

1. **YugabyteDB Anywhere (optional)**, the components that offer a graphical user-interface management portal for creating and administering universes. YugabyteDB Anywhere was formerly referred to as Yugabyte Platform.
2. **Universe**, defines the resource requirements for the underlying database, such as, storage type and size,, number of master (yb-master) and tablet (yb-t-server) servers, and required resources to run, CPU and RAM across the nodes in a cluster,
3. **Database**, running scripts with native DDL, DML appropriate to the underlying database platform (relational or non-relational).

Getting Started

From an operator's machine, clone the following repository.

https://github.com/nycpivot/tanzu-yugabyte

Existing Tanzu Kubernetes Cluster

YugabyteDB Anywhere (optional)

https://docs.yugabyte.com/latest/yugabyte-platform/install-yugabyte-platform/install-software/kubernetes/

01-yugabyte-platform-install.sh

yb-storage.yaml

1. In the terminal window, set the context to your cluster.

   kubectl config use-context [cluster_context]

2. Create the following namespace.

   kubectl create namespace yb-platform

3. Create a secret necessary to run a trial version of the platform.

   cat <<EOF | tee tanzu-yugabyte/yugabyte-k8s-secret.yaml
   apiVersion: v1
   kind: Secret
   metadata:
   name: yugabyte-k8s-pull-secret
   data:
   .dockerconfigjson: $yugabyte_secret
   type: kubernetes.io/dockerconfigjson
   EOF

4. Apply the secret.

   kubectl apply-f tanzu-yugabyte/yugabyte-k8s-secret.yaml -n yb-platform

5. Add the helm chart to the repository and install it on the cluster.

   helm repo add yugabytedb https://charts.yugabyte.com

   helm install yugabyte-platform yugabytedb/yugaware --version 2.13.0 -n yb-platform --wait

6. It's worth checking that all four containers are running in the pod.

   kubectl get pods -n yb-platform

```
ubuntu@ip-172-31-45-151:~$ kubectl get pods -n yb-platform
NAME                            READY   STATUS    RESTARTS   AGE
yugabyte-platform-yugaware-0    4/4     Running   0          52m
```

7. Retrieve the load balancer url hosting the management dashboard and verify it's accessible.

   kubectl get svc -n yb-platform

```
ubuntu@ip-172-31-45-151:~$ kubectl get svc -n yb-platform
NAME                        TYPE           CLUSTER-IP       EXTERNAL-IP                                                              PORT(S)
yugabyte-platform-yugaware-ui   LoadBalancer   100.65.129.116   aa6e53e1d62d547d19f7c571cc7fa4ff-84423801.us-east-2.elb.amazonaws.com   80:30750/TCP,9090:32585/TCP
```

   Navigating to this link will prompt for credentials.

8. Apply the storage class and persistent volumes required for the yb-master and yb-tserver pods. You may want to adjust the number and size of these based on your needs.

   kubectl apply -f tanzu-yugabyte/yb-storage.yaml

9. Confirm both storage class and persistent volumes.

   kubectl get sc

```
ubuntu@ip-172-31-13-200:~$ kubectl get sc
NAME                PROVISIONER                    RECLAIMPOLICY   VOLUMEBINDINGMODE      ALLOWVOLUMEEXPANSION   AGE
default (default)   kubernetes.io/aws-ebs          Delete          Immediate              true                   3h17m
yugabyte-data       kubernetes.io/no-provisioner   Delete          WaitForFirstConsumer   false                  24m
```

   kubectl get pv

```
ubuntu@ip-172-31-13-200:~$ kubectl get pv
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM                                          STORAGECLASS    REASON   AGE
master-disk-1                              5Gi        RWO            Retain           Available                                                  yugabyte-data            27m
master-disk-2                              5Gi        RWO            Retain           Available                                                  yugabyte-data            27m
master-disk-3                              5Gi        RWO            Retain           Available                                                  yugabyte-data            27m
master-disk-4                              5Gi        RWO            Retain           Available                                                  yugabyte-data            27m
master-disk-5                              5Gi        RWO            Retain           Available                                                  yugabyte-data            27m
master-disk-6                              5Gi        RWO            Retain           Available                                                  yugabyte-data            27m
pvc-96ea8fd3-421b-474e-8b63-39ff62189ff4   100Gi      RWO            Delete           Bound       yb-platform/yugabyte-platform-yugaware-storage  default                  29m
tserver-disk-1                             10Gi       RWO            Retain           Available                                                  yugabyte-data            27m
tserver-disk-2                             10Gi       RWO            Retain           Available                                                  yugabyte-data            27m
tserver-disk-3                             10Gi       RWO            Retain           Available                                                  yugabyte-data            27m
tserver-disk-4                             10Gi       RWO            Retain           Available                                                  yugabyte-data            27m
tserver-disk-5                             10Gi       RWO            Retain           Available                                                  yugabyte-data            27m
tserver-disk-6                             10Gi       RWO            Retain           Available                                                  yugabyte-data            27m
```

Configuring YugabyteDB Anywhere

Once logged in to the platform, follow these steps to configure a cluster to deploy Universes.

1. **Configuration**, upload the kubeconfig and secret from the last section to the cluster, define regions and zones.
2. **Universe**, attaches a configuration created in step 1, defines the topology in the cluster for a database, including the number and distribution of pods, their replication factor, enable/disable supported modes (PostgresQL and/or Cassandra), YSQL and/or YCQL, respectively, and security related settings for these endpoints.

Once a Universe is created and configured, it will create a namespace in the cluster of the same name. The following screenshot shows the yb-masters load balancer service, and the yb-tserver load balancer service.

```
ubuntu@ip-172-31-45-151:~$ kubectl get svc -n yb-dev-tanzu-yugabyte-development
NAME                TYPE           CLUSTER-IP      EXTERNAL-IP                                                                PORT(S)
yb-master-service   LoadBalancer   100.67.252.35   aec701fdb27c64ef19f8f194d22cb24e-1398429538.us-east-2.elb.amazonaws.com   7000:31598/TCP
yb-masters          ClusterIP      None            <none>                                                                     7000/TCP,7100/TCP
yb-tserver-service  LoadBalancer   100.66.27.248   a7ab444eaf15f4d32be42b7af2a8952f-1763461590.us-east-2.elb.amazonaws.com   9042:30153/TCP,6379:32542/TCP,5433:30197/TCP
yb-tservers         ClusterIP      None            <none>                                                                     9000/TCP,12000/TCP,11000/TCP,13000/TCP,9100/TCP,6379/TCP,9042/TCP,5433/TCP
```

The external IP and port of the yb-tserver service will be used for executing DDL and DML scripts with the corresponding CLI (ysqlsh or ycqlsh).

NOTE: The next section outlines the configuration of a universe in shell script using helm, with the same result of the above section "Configuring YugabyteDB Anywhere".

Universe

This section creates the environment and resources in which your database will run.

02-yugabyte-universe-create.sh (single availability zone)

03-yugabyte-multiverse-create.sh (multi availability zones - preferred)

1. Execute the script to generate the universe. The script defines the resources you want the universe to use.
2. The output from running this command should list similar results following:

```
ubuntu@ip-172-31-13-200:~$ bash tanzu-yugabyte/03-yugabyte-multiverse-create.sh
namespace/tanzu-yugabyte-multiverse created
WARNING: Kubernetes configuration file is group-readable. This is insecure. Location: /home/ubuntu/.kube/config
WARNING: Kubernetes configuration file is world-readable. This is insecure. Location: /home/ubuntu/.kube/config
NAME: tanzu-yugabyte-multiverse
LAST DEPLOYED: Thu Mar 31 22:08:10 2022
NAMESPACE: tanzu-yugabyte-multiverse
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
1. Get YugabyteDB Pods by running this command:
  kubectl --namespace tanzu-yugabyte-multiverse get pods

2. Get list of YugabyteDB services that are running:
  kubectl --namespace tanzu-yugabyte-multiverse get services

3. Get information about the load balancer services:
  kubectl get svc --namespace tanzu-yugabyte-multiverse

4. Connect to one of the tablet server:
  kubectl exec --namespace tanzu-yugabyte-multiverse -it yb-tserver-0 bash

5. Run YSQL shell from inside of a tablet server:
  kubectl exec --namespace tanzu-yugabyte-multiverse -it yb-tserver-0 -- /home/yugabyte/bin/ysqlsh -h yb-tserver-0.yb-tservers.tanzu-yugabyte-multiverse

6. Cleanup YugabyteDB Pods
  For helm 2:
  helm delete tanzu-yugabyte-multiverse --purge
  For helm 3:
  helm delete tanzu-yugabyte-multiverse -n tanzu-yugabyte-multiverse
  NOTE: You need to manually delete the persistent volume
  kubectl delete pvc --namespace tanzu-yugabyte-multiverse -l app=yb-master
  kubectl delete pvc --namespace tanzu-yugabyte-multiverse -l app=yb-tserver
```

3. Run the first command to ensure the number of pods defined in the above script matches those running.

   kubectl --namespace tanzu-yugabyte-multiverse get pods -o wide

   Confirm the pods are distributed evenly across the nodes. There are affinity rules to prevent pods with the same purpose of being scheduled together. For

example, yb-master–0, yb-master-1, and yb-master-2 should not run together. LIkewise, yb-tserver-0, yb-t-server-1, and yb-tserver-2.

```
ubuntu@ip-172-31-13-200:~$ kubectl --namespace tanzu-yugabyte-multiverse get pods -o wide
NAME           READY   STATUS    RESTARTS   AGE   IP            NODE
yb-master-0    2/2     Running   0          10m   100.96.5.2    ip-10-0-0-60.us-east-2.compute.internal
yb-master-1    2/2     Running   0          10m   100.96.1.3    ip-10-0-4-28.us-east-2.compute.internal
yb-master-2    2/2     Running   0          10m   100.96.2.8    ip-10-0-0-167.us-east-2.compute.internal
yb-tserver-0   2/2     Running   0          10m   100.96.1.2    ip-10-0-4-28.us-east-2.compute.internal
yb-tserver-1   2/2     Running   0          10m   100.96.5.3    ip-10-0-0-60.us-east-2.compute.internal
yb-tserver-2   2/2     Running   0          10m   100.96.2.7    ip-10-0-0-167.us-east-2.compute.internal
```

4. Run the following command to list the services that are used to open the master and tablet dashboards, and the endpoints and ports used for communicating with their respective relational and/or non relational databases.

   kubectl get svc --namespace tanzu-yugabyte-multiverse

```
ubuntu@ip-172-31-13-200:~$ kubectl get svc --namespace tanzu-yugabyte-multiverse
NAME                TYPE           CLUSTER-IP       EXTERNAL-IP                                                                PORT(S)
yb-master-ui        LoadBalancer   100.66.195.107   ac6396a6f765340c5b827074d84ed2ac-834802609.us-east-2.elb.amazonaws.com     7000:31722/TCP
yb-masters          ClusterIP      None             <none>                                                                     7000/TCP,7100/TCP
yb-tserver-service  LoadBalancer   100.64.103.84    a4de638beae294a5e843c38b35ff2633-921740351.us-east-2.elb.amazonaws.com     6379:30549/TCP,9042:30819/TCP,5433:31047/TCP
yb-tservers         ClusterIP      None             <none>                                                                     9000/TCP,12000/TCP,11000/TCP,13000/TCP,9100/TCP,6379/TCP,9042/TCP,5433/TCP
```

   The yb-master-ui is exposed via a load balancer at the specified port.

   The yb-tserver-service exposes the underlying databases via their standard ports.

5. Commands 4 & 5 from step 2 above after running the helm install command can be used to exec into a yb-tserver pod and execute DDL and DML commands manually. The pods contain the YSQLSH and YCQLSH CLI tools for this purpose. For more details, consult the docs at: https://docs.yugabyte.com/latest/quick-start/explore/ysql/#kubernetes

   Download the same CLI tools on your operator machine and run the schema scripts part of the demo applications in the next section.

Databases

This section will use the CLI tools packages with YugabyteDB to communicate with the yb-tserver and relevant ports to create databases, schemas, and perform CRUD operations.

Download YugabyteDB

The source of the package run in the following script can be found at the following link.

https://download.yugabyte.com/

tanzu-yugabyte/10-database-prereqs.sh

Create Databases

Run the following script to create table schemas for the yugastore sample application, and a simple .NET load testing application. It also loads sample data for the yugastore application.

tanzu-yugaybyte/11-databases-create.sh

The source code of the yugastore sample application can be downloaded from here.

https://github.com/yugabyte/yugastore-java

The load balancer url of the yb-tserver-service is listed. Copy and paste that into the prompt. The result of the script should have the following output.

CREATE TABLE

CREATE TABLE

This will be followed by the results of sample data being inserted. A lot of errors resembling the following will be encountered similar to the following.

```
Trouble parsing : Cannot read the array length because "row" is null
java.lang.NullPointerException: Cannot read the array length because "row" is null
        at com.datastax.loader.parser.SetParser.parseIt(SetParser.java:75)
        at com.datastax.loader.parser.AbstractParser.parse(AbstractParser.java:37)
        at com.datastax.loader.parser.DelimParser.parse(DelimParser.java:141)
        at com.datastax.loader.parser.DelimParser.parseWithUnivocity(DelimParser.java:123)
        at com.datastax.loader.parser.DelimParser.parse(DelimParser.java:118)
        at com.datastax.loader.CqlDelimParser.parse(CqlDelimParser.java:365)
        at com.datastax.loader.CqlDelimLoadTask.execute(CqlDelimLoadTask.java:318)
        at com.datastax.loader.CqlDelimLoadTask.call(CqlDelimLoadTask.java:153)
        at com.datastax.loader.CqlDelimLoadTask.call(CqlDelimLoadTask.java:53)
        at java.base/java.util.concurrent.FutureTask.run(FutureTask.java:264)
        at java.base/java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1136)
        at java.base/java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:635)
        at java.base/java.lang.Thread.run(Thread.java:833)
```

Nevertheless, the overall success of the script that data has been inserted depends on the results resembling the following.

```
*** DONE: cronos_products.csv  number of lines processed: 6276 (6276 inserted)
Lines Processed:        6275  Rate:     1255.0
*** Processing cronos_product_rankings.csv
*** DONE: cronos_product_rankings.csv  number of lines processed: 6022 (6022 inserted)
Lines Processed:        6021  Rate:     3010.5
*** Processing cronos_product_inventory.csv
*** DONE: cronos_product_inventory.csv  number of lines processed: 6276 (6276 inserted)
Lines Processed:        6275  Rate:     3137.5
```

Sample Apps

There are two sample applications.

- **Yugastore**, this is built and maintained by Yugabyte. It is a java application that communicates with both supported backend databases, PostgresQL and Cassandra.
- **.NET**, this app continuously writes and reads records to and from an employee PostgresQL table. It is a simple means of observing the resiliency of YugabyteDB when pods, nodes, and entire availability zones are brought offline.