Perforce Helix Core on AWS

Replica Server Deployment Guide

Zenta Hori – Solutions Architect, AWS Game Tech Japan

Kevin Ashman – Senior Partner Solution Architect, Game Tech

December 2020

Introduction

This guide steps you through configuring a Perforce Replica server that has been deployed on AWS following the steps from the AWS Perforce GitHub Sample and the Perforce on AWS Getting Started Blog.

Prerequisites

- You need to deploy your Perforce CloudFormation stack with the "Enable Replica" set to Yes to deploy the replica server.
- Create a S3 bucket with any name in your preferred region. Refer to this instruction. S3 bucket name must be globally unique.
    - For example, the following S3 bucket name can likely avoid collision with other bucket names:
        - (Example) perforce-test-[your name]-[date]

Setting Up Replica Server

To setup replica, you need to configure the primary server first before setting up the replica server with running commands "**p4 configure set**" to write configuration in db.config file of the primary server. This is because the setting for the primary server is replicated to the replica server.

Primary Server Setup

1. First, login to the primary server with ssh. Replace ssh key name and EIP address below with your own.

```
$ ssh -i ~/.ssh/general-key.pem p4admin@54.248.42.141
```

2. If you didn't create a p4admin user previously, you can create one now. If you already created it, you can skip this step and move on to the next step. Run the following command to open vi editor. Check the content and type ":wq" to save it.

```
$ p4 user p4admin
User p4admin saved.
```

3. Run the following command to write a unique server ID "**p4d-ha-awsnva**" to replica server. Note that the server ID of the primary server is already set to "**master.1**".

```
$ p4 server p4d-ha-awsnva
Server p4d-ha-awsnva saved.
```

- vi editor launches when you run the command above. Set "Services:" and "Description:" as shown below, and then save it.

```
# After vi opens, type "i" and modify the content. Once
you're done, press [esc], and then save the changes with
":wq" to close vi.
# Modify "Services:" and "Description:" as shown below
ServerID:    p4d-ha-awsnva
Type:    server
Services:    replica
Options:    nomandatory
Description:   Read-only replica pointing to
master.1:1666
```

4. Run the following command to name the prefix of the output journal file as p4_1.

```
$ p4 configure set master#journalPrefix=/p4/1/checkpoints/p4_1
For server 'master', configuration variable 'journalPrefix' set
to '/p4/1/checkpoints/p4_1'
```

5. Run the following command to set the server named **p4d-ha-awsnva** as the replica server, and point it to 10.0.0.63:1666 as the primary server to synchronize with. This causes the replica to retrieve metadata and version files from the primary server and synchronize with it.

```
$ p4 configure set p4d-ha-awsnva#P4TARGET=10.0.0.63:1666
```

```
For server 'p4d-ha-awsnva', configuration variable 'P4TARGET'
set to '10.0.0.63:1666'
```

6. Here, you create a replication process for the replica. You should setup the following five startup.n commands (if you want to pass multiple space separated values, you must enclose the entire set of values with double quotes). The following commands set p4 pull to poll journal and archive data every one second.

```
$ p4 configure set "p4d-ha-awsnva#startup.1=pull -i 1"
For server 'p4d-ha-awsnva', configuration variable 'startup.1'
set to 'pull -i 1'
```

7. The commands from startup.2 to startup.5 are the same, so you can run it as a loop process as below.

```
$ for i in `seq 2 5`; do p4 configure set "p4d-ha-
awsnva#startup.${i}=pull -u -i 1"; done
For server 'p4d-ha-awsnva', configuration variable 'startup.2'
set to 'pull -u -i 1'
For server 'p4d-ha-awsnva', configuration variable 'startup.3'
set to 'pull -u -i 1'
For server 'p4d-ha-awsnva', configuration variable 'startup.4'
set to 'pull -u -i 1'
For server 'p4d-ha-awsnva', configuration variable 'startup.5'
set to 'pull -u -i 1'
```

8. Run the following commands to set db.replication (metadata permission) and lbr.replication (dipot file permission) to read-only mode.

```
$ p4 configure set "p4d-ha-awsnva#db.replication=readonly"
For server 'p4d-ha-awsnva', configuration variable
'db.replication' set to 'readonly'
```

```
$ p4 configure set "p4d-ha-awsnva#lbr.replication=readonly"
```

```
 For server 'p4d-ha-awsnva', configuration variable 'lbr.replication'
set to 'readonly'
```

9. Setup authentication for server to server communication to allow the replica to connect to the primary server. Set serviceUser to "**svc_replica**" to allow connection. **Service User** is a special user used for authentication.

```
$ p4 configure set p4d-ha-awsnva#serviceUser=svc_replica
For server 'p4d-ha-awsnva', configuration variable 'serviceUser'
set to 'svc_replica'
```

10. Run the following command to create the actual service user "**svc_replica**".

```
$ p4 user -f svc_replica
User svc_replica saved.
```

- After vi editor opens, confirm the following setting, and then type ":wq" to save it.

```
User:   svc_replica

Email:  svc_replica@primary

FullName:       svc_replica
```

11. Set the password for the service user. You can set any password. Just remember it as you'll need it later in the steps.

```
$ p4 passwd svc_replica
Enter new password:
Re-enter new password:
Password updated.
```

12. Create a user group named ServiceUsers.

```
$ p4 group ServiceUsers
Group ServiceUsers created.
```

- After vi editor opens, change the values for “Timeout:” and “Users:” as shown below, and then type “:wq” to save it.

```
# Change the values for Timeout: and Users: as shown below.

Group:  ServiceUsers

MaxResults:     unset

MaxScanRows:    unset

MaxLockTime:    unset

MaxOpenFiles:   unset

Timeout:        unlimited

PasswordTimeout:        unset

Subgroups:

Owners:

Users:  svc_replica
```

13. Grant super privileges to ServiceUsers with p4 protect command.

```
$ p4 protect
Protections saved.
```

- After vi editor opens, add a line “**super group ServiceUsers*//...**” to the last line of Protection: as shown below, and then type “:wq” to save it.

```
# Add “super group ServiceUsers * //...” to the last line
Protections:
      write user * * //...
```

```
        super user p4admin * //...
        super group ServiceUsers * //...
```

14. Run the following command to create a checkpoint on the primary server. The replica server will be built by restoring the checkpoint.

```
$ p4 admin checkpoint -Z
```

15. Create a checkpoint and journal file by running the following command.

```
$ p4 journals -F type=checkpoint -m1 -T jfile
... jfile /p4/1/checkpoints/p4_1.ckp.1.gz
```

16. As you can see with ls command, a current checkpoint has been created with a name “p4_1.ckp.1.gz” as shown below. You can start the replication to apply it to replica server and start it.

```
$ ls -al /p4/1/checkpoints/
合計 36
drwx------ 2 p4admin perforce    67  4 月 24 04:25 .
drwx------ 4 p4admin perforce    39  4 月 23 11:09 ..
-r--r----- 1 p4admin perforce  3703  4 月 24 04:25 p4_1.ckp.1.gz
-r--r----- 1 p4admin perforce    70  4 月 24 04:25 p4_1.ckp.1.md5
-r--r----- 1 p4admin perforce 27398  4 月 24 04:25 p4_1.jnl.0
```

17. Copy the checkpoint file to S3 bucket you already created to use it in replica. Replace <YOUR BUCKET> in the command below with your bucket name (e.g. perforce-test-[your name]-[date]) you created earlier in the “Prerequisites” section.

```
$ aws s3 cp /p4/1/checkpoints/p4_1.ckp.1.gz s3://<YOUR
BUCKET>/aws-perforce/checkpoints/
upload: ../../p4/1/checkpoints/p4_1.ckp.1.gz to s3://<YOUR
BUCKET>/aws-perforce/checkpoints/p4_1.ckp.1.gz
```

18. Copy the depot data from the primary server to S3 bucket to use it in replica as well.

```
$ aws s3 cp  /p4/1/depots/  s3://<YOUR BUCKET>/aws-
perforce/depots/ --recursive
upload: ../../p4/1/depots/spec/protect.p4s,d/1.1.gz to s3://<YOUR
BUCKET>/aws-perforce/depots/spec/protect.p4s,d/1.1.gz
upload: ../../p4/1/depots/spec/user/71,d/p4admin.p4s,d/1.1.gz to
s3://<YOUR BUCKET>/aws-
perforce/depots/spec/user/71,d/p4admin.p4s,d/1.1.gz
upload: ../../p4/1/depots/spec/group/12,d/serviceusers.p4s,d/1.1.
gz to s3://<YOUR BUCKET>/aws-
perforce/depots/spec/group/12,d/serviceusers.p4s,d/1.1.gz
upload: ../../p4/1/depots/spec/server/74,d/p4d-ha-
awsnva.p4s,d/1.1.gz to s3://<YOUR BUCKET>/aws-
perforce/depots/spec/server/74,d/p4d-ha-awsnva.p4s,d/1.1.gz
upload: ../../p4/1/depots/spec/user/96,d/svc_replica.p4s,d/1.1.gz
to s3://<YOUR BUCKET>/aws-
perforce/depots/spec/user/96,d/svc_replica.p4s,d/1.1.gz
upload: ../../p4/1/depots/spec/server/48,d/replica1.p4s,d/1.1.gz
to s3://<YOUR BUCKET>/aws-
perforce/depots/spec/server/48,d/replica1.p4s,d/1.1.gz
upload: ../../p4/1/depots/spec/depot/7,d/unload.p4s,d/1.1.gz to
s3://<YOUR BUCKET>/aws-
perforce/depots/spec/depot/7,d/unload.p4s,d/1.1.gz
```

19. We finished the steps on the primary server. Next let's move on to the replica server to set up it.

Replica Server Setup

1. Login to the replica server via SSH to start setup. Replace SSH key name and EIP address below with your own. You can find the IP address for the replica server at **PerforceReplicaEIP** on **Outputs** tab in the CloudFormation stack.

```
$ ssh -i ~/.ssh/general-key.pem p4admin@18.180.134.170
```

2. Set the environment variable to memorize connection destination just as you did for the primary server.

```
$ echo 'export P4PORT=localhost:1666' >> ~/.bash_profile
$ source ~/.bash_profile
```

3. Now copy the checkpoint file and the depot you just saved in S3 to the replica server. Replace <YOUR BUCKET> in the command below with your bucket name (e.g. perforce-test-[your name]-[date]) you created in the "Prerequisites" section.

```
$ aws s3 cp s3://<YOUR BUCKET>/aws-perforce/checkpoints/p4_1.ckp.1.gz
/p4/1/checkpoints/
download: s3://<YOUR BUCKET>/aws-perforce/checkpoints/p4_1.ckp.1.gz
to ../../p4/1/checkpoints/p4_1.ckp.1.gz
```

```
$ aws s3 cp s3://<YOUR BUCKET>/aws-perforce/depots/ /p4/1/depots/ --
recursive
download: s3://<YOUR BUCKET>/aws-
perforce/depots/spec/protect.p4s,d/1.1.gz
to ../../p4/1/depots/spec/protect.p4s,d/1.1.gz
download: s3://<YOUR BUCKET>/aws-
perforce/depots/spec/depot/7,d/unload.p4s,d/1.1.gz
to ../../p4/1/depots/spec/depot/7,d/unload.p4s,d/1.1.gz
download: s3://<YOUR BUCKET>/aws-
perforce/depots/spec/group/12,d/serviceusers.p4s,d/1.1.gz
to ../../p4/1/depots/spec/group/12,d/serviceusers.p4s,d/1.1.gz
download: s3://<YOUR BUCKET>/aws-perforce/depots/spec/server/74,d/p4d-
ha-awsnva.p4s,d/1.1.gz to ../../p4/1/depots/spec/server/74,d/p4d-ha-
awsnva.p4s,d/1.1.gz
download: s3://<YOUR BUCKET>/aws-
perforce/depots/spec/user/71,d/p4admin.p4s,d/1.1.gz
to ../../p4/1/depots/spec/user/71,d/p4admin.p4s,d/1.1.gz
download: s3://<YOUR BUCKET>/aws-
perforce/depots/spec/server/48,d/replica1.p4s,d/1.1.gz
to ../../p4/1/depots/spec/server/48,d/replica1.p4s,d/1.1.gz
```

```
download: s3://<YOUR BUCKET>/aws-
perforce/depots/spec/user/96,d/svc_replica.p4s,d/1.1.gz
to ../../p4/1/depots/spec/user/96,d/svc_replica.p4s,d/1.1.gz
```

4. Stop p4d process of the replica to apply the copied checkpoint to the replica.

```
$ /p4/1/bin/p4d_1_init stop
```

5. Delete all old database files still remaining in the replica.

```
$ sudo rm -fr /p4/1/root/db.*
```

6. Apply the checkpoint file to the replica to restore the database.

```
$ /p4/1/bin/p4d_1 -r /p4/1/root -jr -z /p4/1/checkpoints/p4_1.ckp.1.gz
Perforce db files in '/p4/1/root' will be created if missing...
Recovering from /p4/1/checkpoints/p4_1.ckp.1.gz...
```

7. Run the following script to setup.

```
$ source /p4/common/bin/p4_vars 1
```

8. Connect to the primary server as the service user you just set up earlier. A password is required and then enter the password you set up earlier in step 11 of “Primary Server Setup”.

```
$ /p4/1/bin/p4_1 -p 10.0.0.63:1666 -u svc_replica login
Enter password:
User svc_replica logged in.
```

9. Now that all setting has been reflected, start the server as the replica.

```
$ /p4/1/bin/p4d_1_init start
Preflight check: /p4/1/bin/p4d_1 -r /p4/1/root -xvU
Preflight journal corruption check
Starting /p4/1/bin/p4d_1 -p 1666 -r /p4/1/root -J /p4/1/logs/journal -L
/p4/1/logs/log -q --daemonsafe --pid-file
```

10. After the server is restarted, execute "**p4 info**" command to check "Server services:replica" and "Replica of: 10.0.0.63:1666" which means that the server is running as the replica.

```
$ p4 info
User name: perforce
Client name: p4d-ha-awsnva
Client host: p4d-ha-awsnva
Client unknown.
Current directory: /home/p4admin
Peer address: 127.0.0.1:43042
Client address: 127.0.0.1
Server address: localhost:1666
Server root: /p4/1/root
Server date: 2020/04/24 05:34:26 +0000 UTC
Server uptime: 00:02:33
Server version: P4D/LINUX26X86_64/2019.2/1942501 (2020/04/02)
ServerID: p4d-ha-awsnva
Server services: replica
Replica of: 10.0.0.63:1666
Server license: none
Case Handling: insensitive
```

11. Run the command below to check if p4 pull process is running to periodically synchronize data with the primary server. Running five synchronization processes configured with p4 configure earlier on the primary server means that the replica is successfully synchronized. If you get an error saying "You don't have permission for this operation.", then log out and log in again with ssh, and then run the command again.

```
$ p4 monitor show -a
 5631 B svc_replic 00:05:11 pull sleeping 1000 ms
 5632 B svc_replic 00:05:11 pull sleeping 1000 ms
 5633 B svc_replic 00:05:11 pull sleeping 1000 ms
 5634 B svc_replic 00:05:11 pull sleeping 1000 ms
 5635 B svc_replic 00:05:11 pull sleeping 1000 ms
 5685 R p4admin    00:00:00 monitor show -a
```

12. Run the following command to check the status of replication. If [Current replica journal state] and [Current primary journal state] point to the same position, it means the replica can keep up to date. If the sequence values are different and the replica is behind, then wait for a while and run the command again to confirm it has updated.

```
$ p4 pull -lj
Current replica journal state is:    Journal 1,    Sequence 2315.
Current master journal state is:    Journal 1,    Sequence 2315.
The statefile was last modified at:    2020/04/24 08:10:08.
The replica server time is currently:    2020/04/24 08:11:05 +0000 UTC
```

13. When you copied the versioned files from the primary server to the replica server, you relied on the operating system to transfer the files. To determine whether data was corrupted in the process, run p4 verify on the replica server.

```
$ p4 verify //...
//spec/depot/unload.p4s#1 - add default change (text+C)
F91D33DCE8E82DBA82859BB767392A07
//spec/group/ServiceUsers.p4s#1 - add default change (text+C)
F550D3915DC8B032BA6ED084DA9AA068
//spec/protect.p4s#1 - add default change (text+C)
A0240A13D4785D3F53C0FDE3E029A605
//spec/server/p4d-ha-awsnva.p4s#1 - add default change (text+C)
94350051CA792A37847BC516DE94258D
//spec/server/replica1.p4s#1 - add default change (text+C)
A60AA4A0394FD9213B389E0285B30CDD
//spec/user/p4admin.p4s#1 - add default change (text+C)
EE316B2F92F6E8AD5A49CB653C73D2F9
//spec/user/svc_replica.p4s#1 - add default change (text+C)
A5E99395666E65C044F33D718FB3A5C8
```

We finished the steps to build the replica server. Next we test the infrastructure and replication.

Testing File Replication

Now let's test and see if newly created files on the primary server are replicated to the replica server.

1. Login to the primary server via SSH. Replace SSH key name and EIP address below with your own.

```
$ ssh -i ~/.ssh/general-key.pem p4admin@18.180.250.162
```

2. Move to "**repository**" directory created previously in the advanced configuration guide. Create a new test file named test2.config.

```
$ cd repository/
$ echo test2 > test2.config
```

3. Add the file to a changelist with the following command.

```
$ p4 -c p4admin add -t text test2.config
//repository/test2.config#1 - opened for add
```

4. Submit the change to the depot with the following command.

```
$ p4 -c p4admin submit
Change 1 created with 1 open file(s).
Submitting change 1.
Locking 1 files ...
add //repository/test2.config#1
Change 1 submitted.
```

The command opens vi editor and describe the reason for the change in the "Description:" section.

```
# When vi launches, describe the change. Save the change with ":wq" to
close vi.
# Describe the change
Description:
```

```
        add a new file test2
Files:
        //repository/test2.config      # add
```

5. Next, login to the replica server via SSH. Replace SSH key name and EIP address below with your own.

```
$ ssh -i ~/.ssh/general-key.pem p4admin@18.180.134.170
```

6. Check the synchronization status of the replica. You can see that the sequence number is bigger than before.

```
   $ p4 pull -lj
   Current replica journal state is:    Journal 1,    Sequence 6352.
   Current master journal state is:    Journal 1,    Sequence 6352.
   The statefile was last modified at:    2020/06/04 07:45:45.
   The replica server time is currently:    2020/06/04 07:54:39 +0000
   UTC
```

7. Run p4 print command to see the replicated files and the contents on the replica server. The latest addition to the file is shown as “test2” properly.

```
$ p4 print //repository/test2.config
//repository/test2.config#1 - add change 1 (text)
test2
```

8. If you want to save the file in the local disk, run the following command.

```
$ p4 print -o test2.config //repository/test2.config
//repository/test2.config#1 - add change 1 (text)
```

Summary

You have now configured and verified your Perforce replica server. Your infrastructure is now Highly Available, allowing for your primary server to become unhealthy or unavailable and allowing users to use the replica server. Below you will

find additional references that will help you further configure your replica and customize your CloudFormation template for your own needs.

REFERENCES

- Configuring a read-only replica
- Release and license information: adding or updating
- Securing Perforce server
- Enabling SSL support
- Source code for AWS CloudFormation template to build Perforce on AWS