3

Getting Started with CLI

ORACLE®

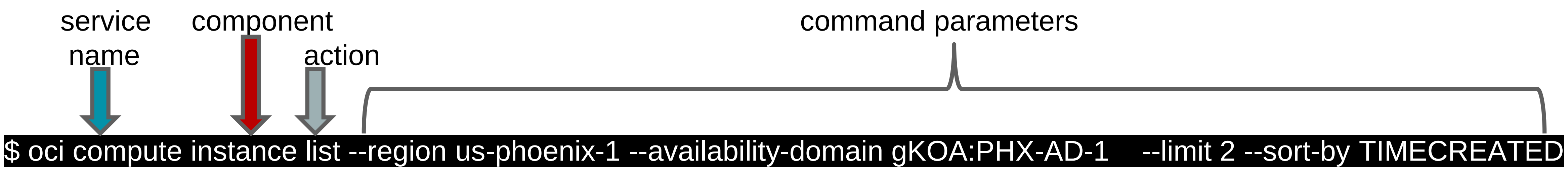
Agenda

In this module we will review the OCI Command Line Interface (CLI). Upon completion, you should be able to:

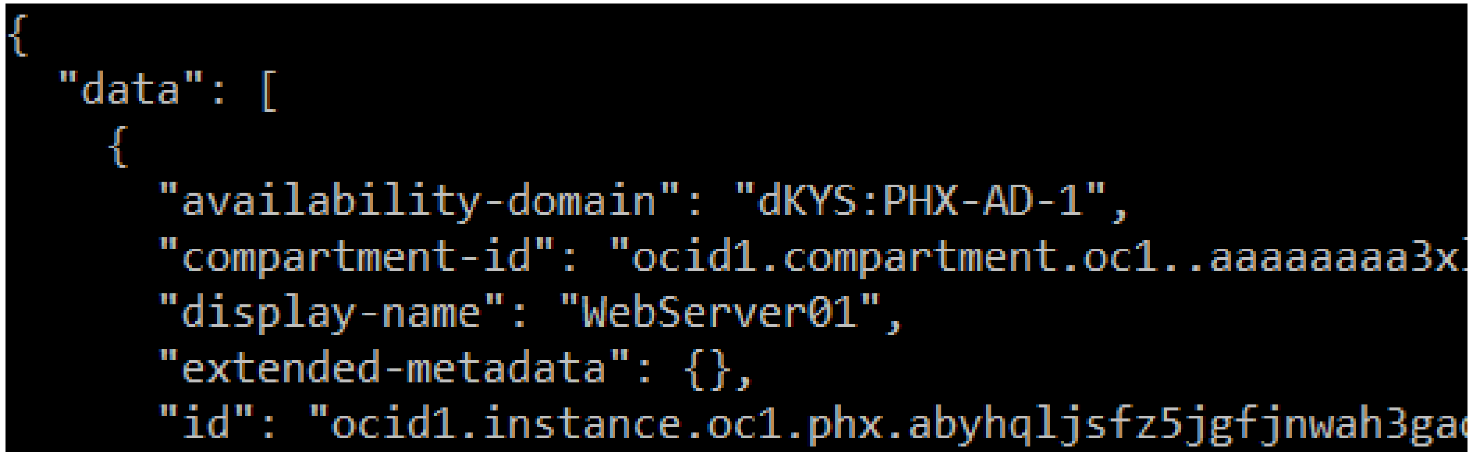
- Describe key features and functionality OCI CLI
- Understand how to use the CLI to create, discover, and terminate resources
- Create shell scripts utilizing the CLI to make your life better

CLI Command Construct

The OCI CLI is a unified tool that allows interaction with most service through a single command. After entering the program command specify the service, the action, and any additional switches.



```
$ oci compute instance list --region us-phoenix-1 --availability-domain gKOA:PHX-AD-1    --limit 2 --sort-by TIMECREATED
```



```
{
  "data": [
    {
      "availability-domain": "dKYS:PHX-AD-1",
      "compartment-id": "ocid1.compartment.oc1..aaaaaaaa3x
      "display-name": "WebServer01",
      "extended-metadata": {},
      "id": "ocid1.instance.oc1.phx.abyhqljsfz5jgfjnwah3ga
```

Configure the CLI

Before using the CLI, you must provide IAM credentials with appropriate access. The credentials provided will be separated into profiles and stored in ~/.oci/config

Credentials may either be input manually, or by using the setup command:

```
$ oci setup config
```

Sample config file

```
$ cat ~/.oci/config
[DEFAULT]
user=ocid1.user.oc1..aaaaaaaaja3b5xxxxlk235lwlndbm6mf7pznq
fingerprint=8b:cc:yy:yy:yy:84:5f:ee:b4
key_file=/home/opc/.oci/oci_api_key.pem
tenancy=ocid1.tenancy.oc1..aaaaaaaaxy6bh46cdnwer23cxtfxhhva2hna
region=us-phoenix-1
```

CLI Profiles

Profiles allow you to specific multiple sets of credentials within your config file. Once defined, you may reference the appropriate profile at runtime.

```
[DEFAULT]
…
…
[dev_compartment]
user=ocid1.user.oc1..aaaaaaaaja3b5st7wdf9sdfdhwlndbm6mf7pznq
fingerprint=8b:cc:27:61:dd:yy:yy:yy:yy:yy:84:5f:ee:b4
key_file=/home/opc/.oci/oci_api_key.pem
tenancy=ocid1.tenancy.oc1..aaaaaaaaxy63cxtfxhhva2hna
region=us-phoenix-1
```

```
$ oci compute image list --profile dev_compartment
```

CLI RC shortcuts

- The **oci_cli_rc** creates shortcuts and command abbreviations.
- Can be used to set default compartment per CLI profile
- Setup is simple:
  ```
  $ oci setup oci-cli-rc
  ```

- Some of the section included:
  ```
  $ cat ~/.oci/oci_cli_rc
  [OCI_CLI_CANNED_QUERIES
  [OCI_CLI_COMMAND_ALIASES]
  rm = os.object.delete
  [OCI_CLI_PARAM_ALIASES]
  --cid = --compartment-id
  ```

Helpful Features

Working with the CLI can allow for streamlined operations through automation. The following features can reduce the effort of command execution and simplify the resulting command output.

--output [json | table] – alter the format of the command output

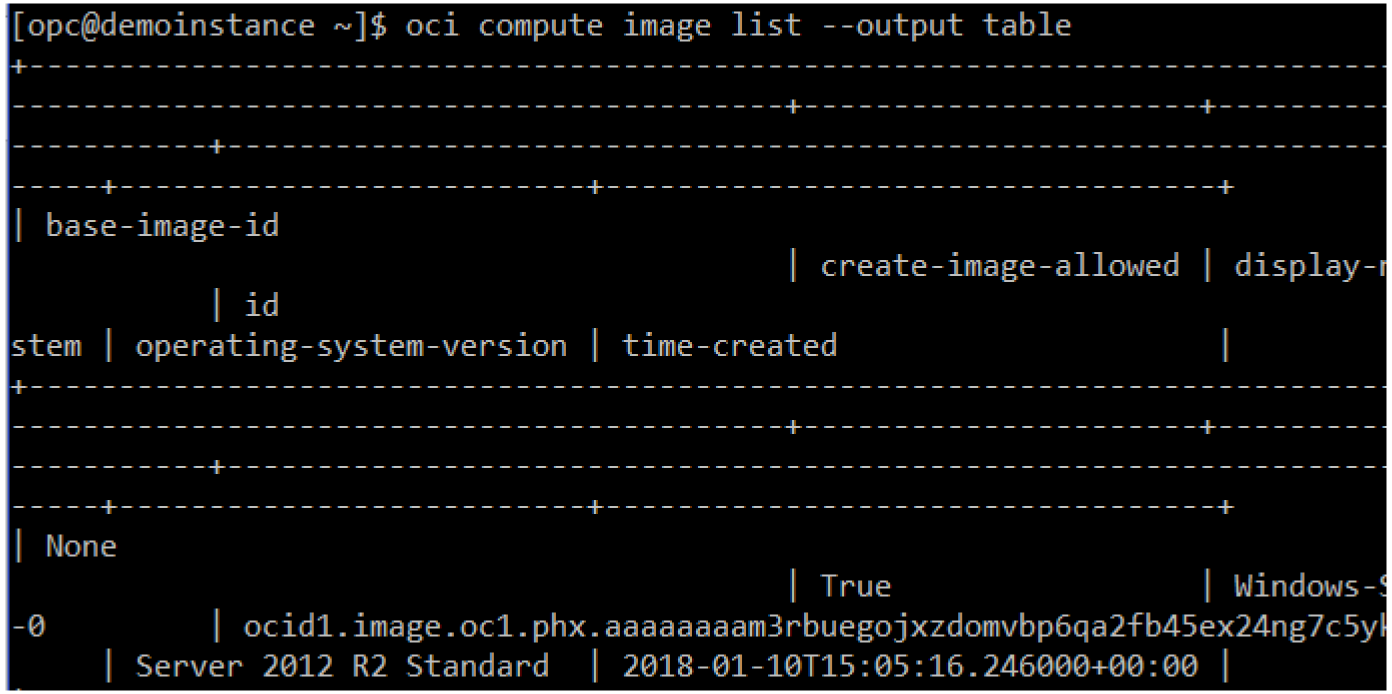
--query – allows input of a JMESPath query to refine the command output

--generate-full-command-json-input – Prints out a JSON document containing all available options for the specified command.

--from-json – will consume the parameters as defined in the file created by --generate-full-command-json-input.

Using the --query option

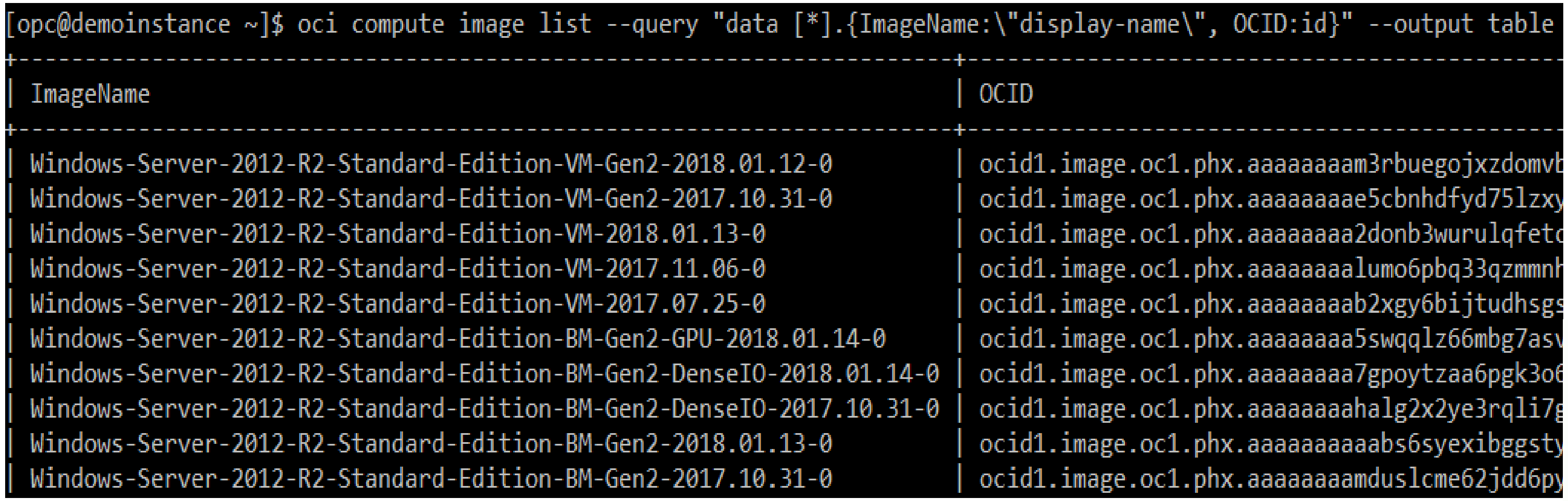
```
$ oci compute image list --output table
```



```
[opc@demoinstance ~]$ oci compute image list --output table
+---------------------------------------------------------------------------
-------------------------------------------+----------------------+---------
-----------+----------------------------------------------------------------
-----+--------------------------+----------------------------------+
| base-image-id
                                           | create-image-allowed | display-
           | id
stem | operating-system-version | time-created                     |
+---------------------------------------------------------------------------
-------------------------------------------+----------------------+---------
-----------+----------------------------------------------------------------
-----+--------------------------+----------------------------------+
| None
                                           | True                 | Windows-
-0         | ocid1.image.oc1.phx.aaaaaaaam3rbuegojxzdomvbp6qa2fb45ex24ng7c5y
     | Server 2012 R2 Standard  | 2018-01-10T15:05:16.246000+00:00 |
```

Using the --query option

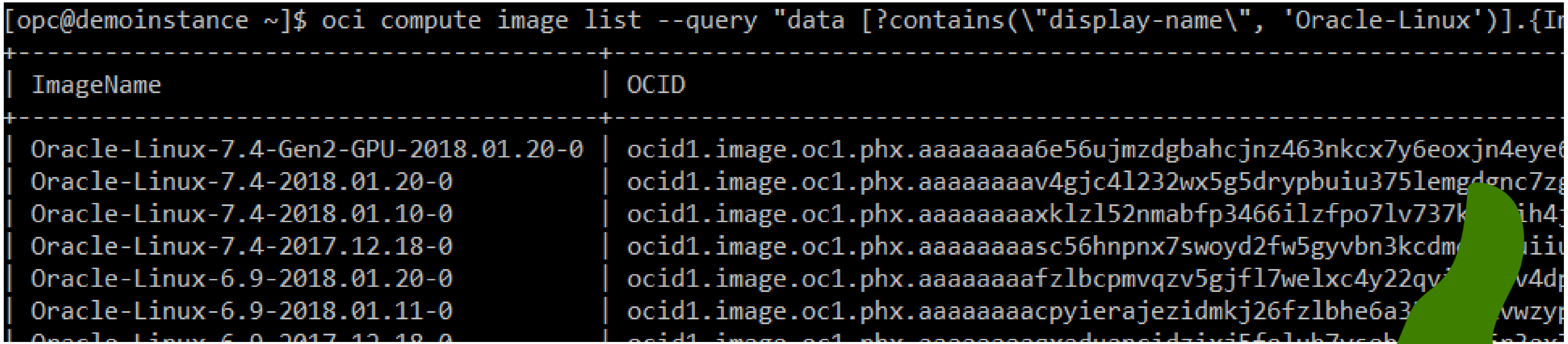
```
$ oci compute image list --query "data [*].{ImageName:\"display-name\", OCID:id}" --output table
```



```
[opc@demoinstance ~]$ oci compute image list --query "data [*].{ImageName:\"display-name\", OCID:id}" --output table
+-----------------------------------------------------------------+----------------------------------------
| ImageName                                                       | OCID
+-----------------------------------------------------------------+----------------------------------------
| Windows-Server-2012-R2-Standard-Edition-VM-Gen2-2018.01.12-0    | ocid1.image.oc1.phx.aaaaaaaam3rbuegojxzdomvb
| Windows-Server-2012-R2-Standard-Edition-VM-Gen2-2017.10.31-0    | ocid1.image.oc1.phx.aaaaaaaae5cbnhdfyd75lzxy
| Windows-Server-2012-R2-Standard-Edition-VM-2018.01.13-0         | ocid1.image.oc1.phx.aaaaaaaa2donb3wurulqfetc
| Windows-Server-2012-R2-Standard-Edition-VM-2017.11.06-0         | ocid1.image.oc1.phx.aaaaaaaalumo6pbq33qzmmnh
| Windows-Server-2012-R2-Standard-Edition-VM-2017.07.25-0         | ocid1.image.oc1.phx.aaaaaaaab2xgy6bijtudhsgs
| Windows-Server-2012-R2-Standard-Edition-BM-Gen2-GPU-2018.01.14-0 | ocid1.image.oc1.phx.aaaaaaaa5swqqlz66mbg7asv
| Windows-Server-2012-R2-Standard-Edition-BM-Gen2-DenseIO-2018.01.14-0 | ocid1.image.oc1.phx.aaaaaaaa7gpoytzaa6pgk3o6
| Windows-Server-2012-R2-Standard-Edition-BM-Gen2-DenseIO-2017.10.31-0 | ocid1.image.oc1.phx.aaaaaaaahalg2x2ye3rqli7g
| Windows-Server-2012-R2-Standard-Edition-BM-Gen2-2018.01.13-0    | ocid1.image.oc1.phx.aaaaaaaaaabs6syexibggsty
| Windows-Server-2012-R2-Standard-Edition-BM-Gen2-2017.10.31-0    | ocid1.image.oc1.phx.aaaaaaaamduslcme62jdd6py
```

Using the --query option with qualifiers

```
$ oci compute image list --query "data [?contains(\"display-name\", 'Oracle-Linux-7.7')].{ImageName:\"display-name\", OCID:id}" --output table
```



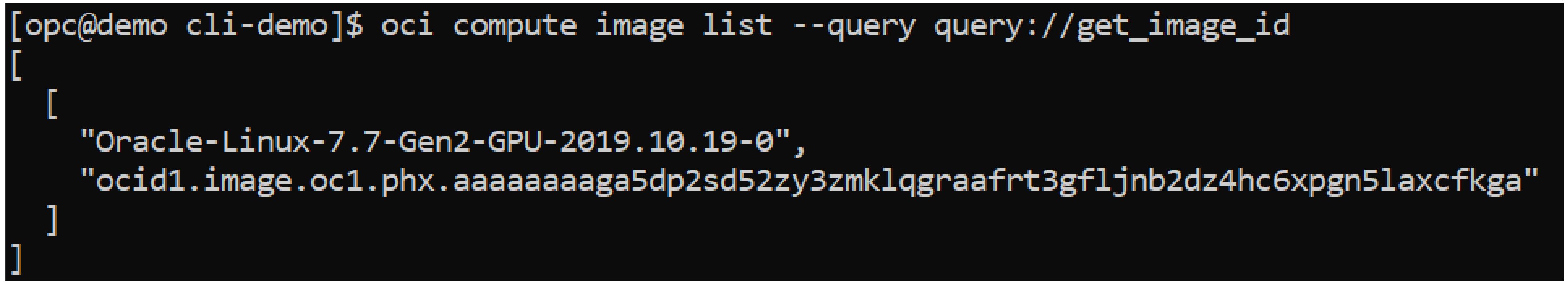
```
[opc@demoinstance ~]$ oci compute image list --query "data [?contains(\"display-name\", 'Oracle-Linux')].{Im
+---------------------------------------+------------------------------------------------------------------
| ImageName                             | OCID
+---------------------------------------+------------------------------------------------------------------
| Oracle-Linux-7.4-Gen2-GPU-2018.01.20-0 | ocid1.image.oc1.phx.aaaaaaaa6e56ujmzdgbahcjnz463nkcx7y6eoxjn4eye
| Oracle-Linux-7.4-2018.01.20-0          | ocid1.image.oc1.phx.aaaaaaaav4gjc4l232wx5g5drypbuiu375lemgdgnc7zg
| Oracle-Linux-7.4-2018.01.10-0          | ocid1.image.oc1.phx.aaaaaaaaxklzl52nmabfp3466ilzfpo7lv737k    ih4j
| Oracle-Linux-7.4-2017.12.18-0          | ocid1.image.oc1.phx.aaaaaaaasc56hnpnx7swoyd2fw5gyvbn3kcdm     uiiu
| Oracle-Linux-6.9-2018.01.20-0          | ocid1.image.oc1.phx.aaaaaaaafzlbcpmvqzv5gjfl7welxc4y22qv      v4dp
| Oracle-Linux-6.9-2018.01.11-0          | ocid1.image.oc1.phx.aaaaaaaacpyierajezidmkj26fzlbhe6a3        vwzyp
```

Saving Canned Queries

- Register in the oci_cli_rc file with a simple name

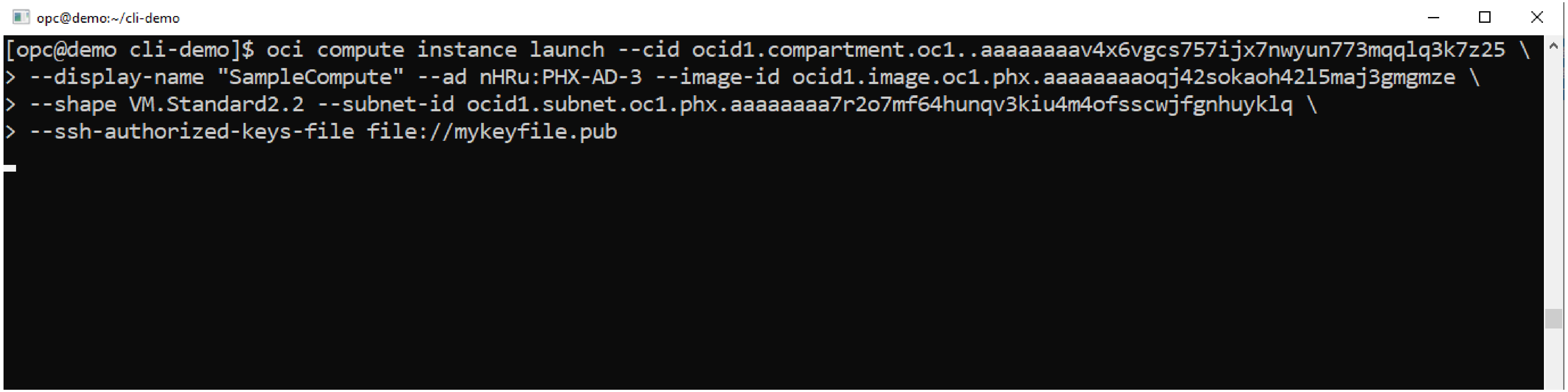
```
[OCI_CLI_CANNED_QUERIES]
get_image_id=reverse(sort_by(data[?contains("display-name", `Oracle-Linux-7.7`)], &"time-created"))|[0:1].["display-name",id]
```

- Use the canned query at the CLI



```
[opc@demo cli-demo]$ oci compute image list --query query://get_image_id
[
  [
    "Oracle-Linux-7.7-Gen2-GPU-2019.10.19-0",
    "ocid1.image.oc1.phx.aaaaaaaaga5dp2sd52zy3zmklqgraafrt3gfljnb2dz4hc6xpgn5laxcfkga"
  ]
]
```

Launch Compute Instance Command



opc@demo:~/cli-demo

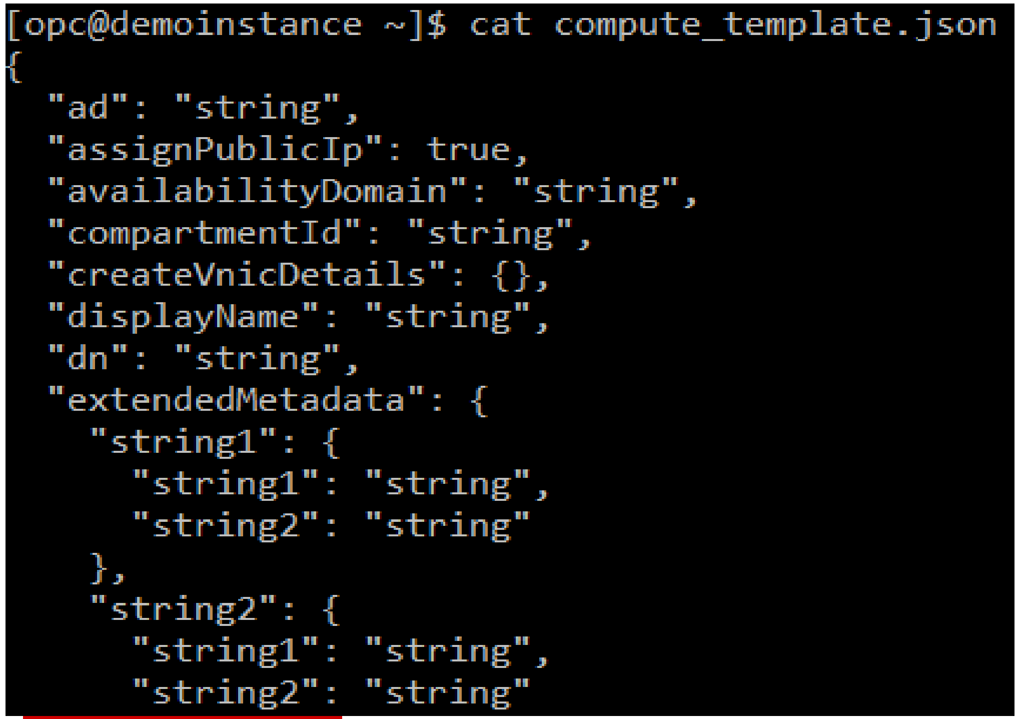
```
[opc@demo cli-demo]$ oci compute instance launch --cid ocid1.compartment.oc1..aaaaaaaav4x6vgcs757ijx7nwyun773mqqlq3k7z25 \
> --display-name "SampleCompute" --ad nHRu:PHX-AD-3 --image-id ocid1.image.oc1.phx.aaaaaaaaoqj42sokaoh42l5maj3gmgmze \
> --shape VM.Standard2.2 --subnet-id ocid1.subnet.oc1.phx.aaaaaaaa7r2o7mf64hunqv3kiu4m4ofsscwjfgnhuyklq \
> --ssh-authorized-keys-file file://mykeyfile.pub
```

This is fine if you are building it into a script. Not so if you are typing it out repeatedly.

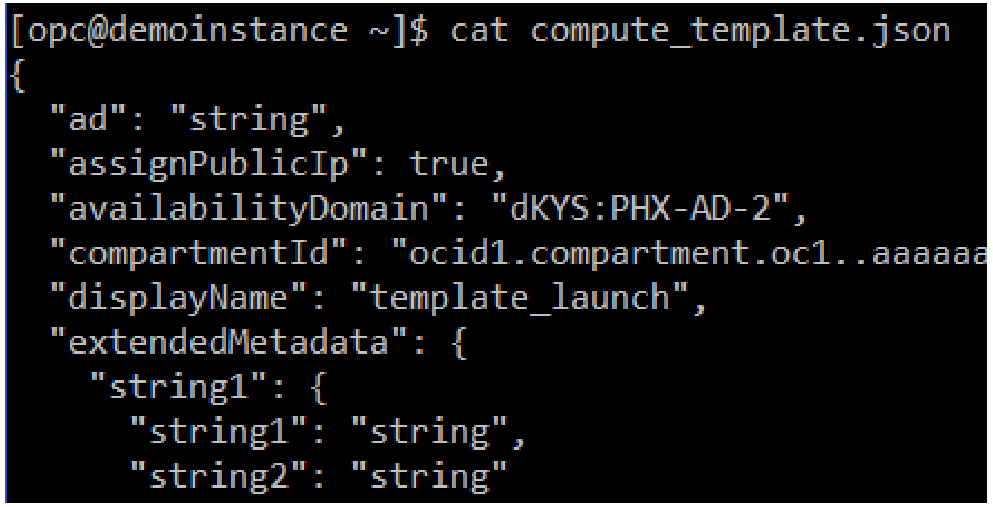
Working with JSON input templates

As we mentioned earlier, the command parameter --generate-full-command-json-input can be used to create a template shell into which you may configure specific values. This is great for creating resource templates such as a development compute instance.

```
$ oci compute instance launch --generate-full-command-json-input > compute_template.json
```



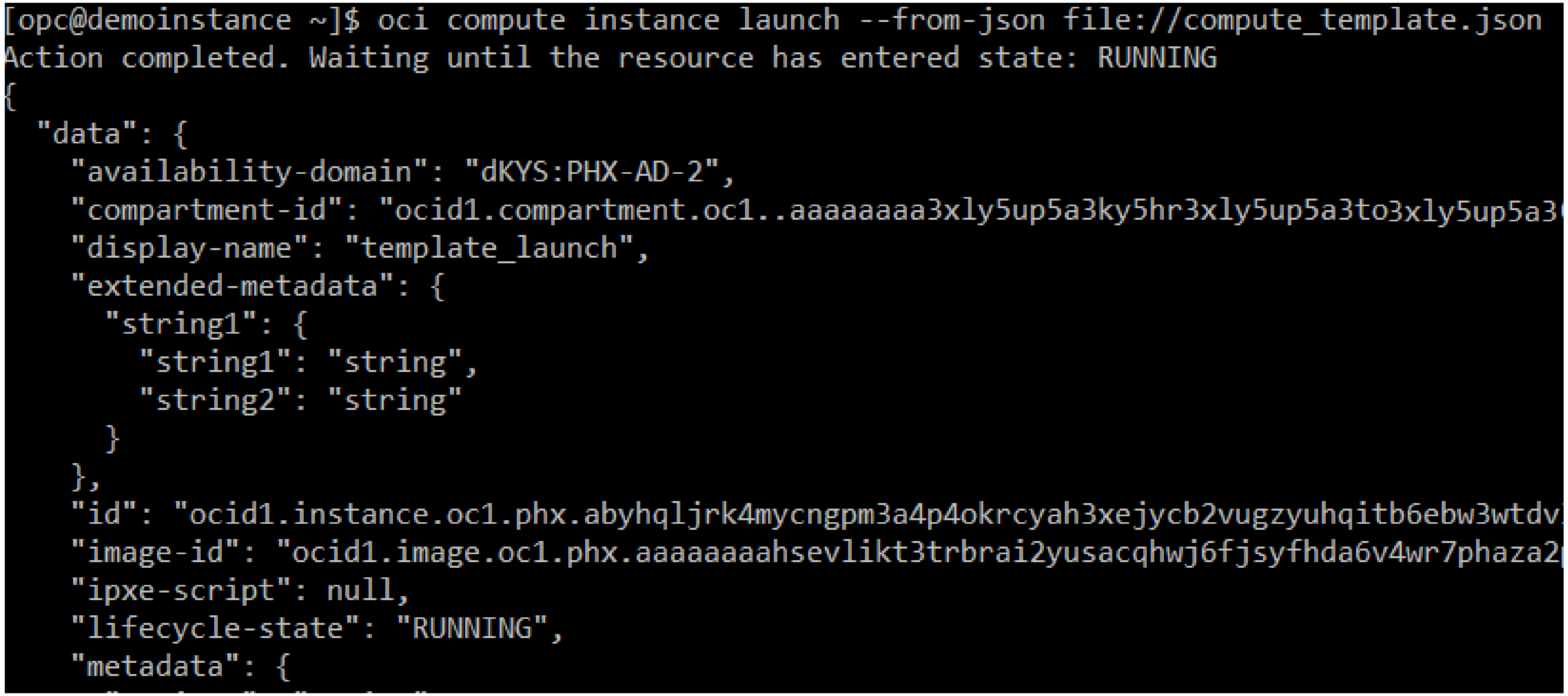
```
[opc@demoinstance ~]$ cat compute_template.json
{
  "ad": "string",
  "assignPublicIp": true,
  "availabilityDomain": "string",
  "compartmentId": "string",
  "createVnicDetails": {},
  "displayName": "string",
  "dn": "string",
  "extendedMetadata": {
    "string1": {
      "string1": "string",
      "string2": "string"
    },
    "string2": {
      "string1": "string",
      "string2": "string"
```



```
[opc@demoinstance ~]$ cat compute_template.json
{
  "ad": "string",
  "assignPublicIp": true,
  "availabilityDomain": "dKYS:PHX-AD-2",
  "compartmentId": "ocid1.compartment.oc1..aaaaaa
  "displayName": "template_launch",
  "extendedMetadata": {
    "string1": {
      "string1": "string",
      "string2": "string"
```

Launch a Compute instance with the JSON template

```
$ oci compute instance launch --from-json file://compute_template.json
```



```
[opc@demoinstance ~]$ oci compute instance launch --from-json file://compute_template.json
Action completed. Waiting until the resource has entered state: RUNNING
{
  "data": {
    "availability-domain": "dKYS:PHX-AD-2",
    "compartment-id": "ocid1.compartment.oc1..aaaaaaaa3xly5up5a3ky5hr3xly5up5a3to3xly5up5a3
    "display-name": "template_launch",
    "extended-metadata": {
      "string1": {
        "string1": "string",
        "string2": "string"
      }
    },
    "id": "ocid1.instance.oc1.phx.abyhqljrk4mycngpm3a4p4okrcyah3xejycb2vugzyuhqitb6ebw3wtdv
    "image-id": "ocid1.image.oc1.phx.aaaaaaaahsevlikt3trbrai2yusacqhwj6fjsyfhda6v4wr7phaza2
    "ipxe-script": null,
    "lifecycle-state": "RUNNING",
    "metadata": {
```

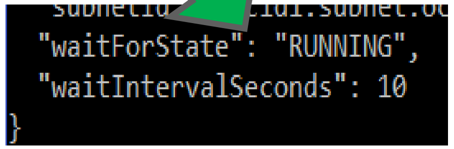
Using a script to orchestrate several tasks

The following is an example of a simple BASH script. In the script we launch a new instance, wait for it to respond to an SSH request, then test the sample website.

```
#!/bin/bash
instance_id=$(oci compute instance launch --from-json file://compute_template.json \
  --query ‘data.id’ --raw-output)
pub_ip=$(oci compute instance list-vnics --instance-id $instance_id --query \
  ‘data[*]|[0].”public-ip”’ --raw-output)

# verify SSH connectivity
ssh -qi ~/.ssh/id_rsa opc@$pub_ip
while [ $? -ne 0 ]; do
  echo "Checking SSH connectivity“ && sleep 10
  ssh -qi ~/.ssh/id_rsa opc@$pub_ip
done
echo "SSH is up - lets move on!“ && sleep 3

#run a simple test
curl http://$pub_ip/testpage.html
```



```
 "waitForState": "RUNNING",
 "waitIntervalSeconds": 10
}
```

Example: List Compute + Network Resources

```
printf "Compartment\tVCN\tSubnet\tAddress\tName\n"
for c in $(oci iam compartment list --compartment-id-in-subtree true --all --raw-
output --query 'join(`\n`, data[*].id)')
do
  for v in $(oci network vcn list --compartment-id $c --raw-output --query
'join(`\n`, data[*].id)')
  do
    for s in $(oci network subnet list --compartment-id $c --vcn-id $v --raw-
output --query 'join(`\n`, data[*].id)')
    do
      oci network private-ip list --subnet-id $s --raw-output --query
"data[*].join(\`\\t\`, [\`$c\`, \`$v\`, \`$s\`, \"ip-address\", \"display-
name\"]) | join(\`\\n\`, @)"
    done
  done
done
```

Example: Output

```
[opc@demo oci-cli]$ ./test.sh
Compartment     VCN     Subnet  Address Name
ocid1.compartment.. ocid1.vcn.. ocid1.subnet.. 10.0.1.2        ATPLab
ocid1.compartment.. ocid1.vcn.. ocid1.subnet.. 10.0.0.2        WebServer
ocid1.compartment.. ocid1.vcn.. ocid1.subnet.. 192.168.0.9     MySqlDB
```

Scenario: Copy to/from Object Storage

Moving large amounts of data to or from Object Storage can be simplified and automated using the CLI. Think data backups, log rotation, or simple archival tasks.

With the CLI:

- Automatically use multi-part downloads (GET) for faster transfer. You can, however, control the chunk size and parallelism:
  ```
  $ oci os object get –bn MyBucket --file My10Gfile  --name MyObject --part-size 1024 –multipart-download-threshold 1024
  ```
- Automatically use multi-part for uploads (PUT). You can control the parameters, or disable it if you choose:
  ```
  $ oci os object put --bn MyBucket --file My10Gfile --part-size 512 --parallel-upload-count 5

  $ oci os object put --bn MyBucket --file My10Gfile --no-multipart
  ```

Knowledge Check

As part of your daily tasks, you deploy 5-10 development instances on OCI for the engineering team. All instances have the same configuration parameters. What CLI feature could you use to simplify the deployment process?

A. Use the CLI to `generate-full-command-json-input` and fill in the details of the template.

B. Download and use a macro recording tool for your browser. Record the steps to log in and deploy the instance.

C. Use the command `oci compute instance launch --use-last-configuration`

D. Use the `--query` feature

Knowledge Check

As part of your daily tasks, you deploy 5-10 development instances on OCI for the engineering team. All instances have the same configuration parameters. What CLI feature could you use to simplify the deployment process?

A. Use the CLI to `generate-full-command-json-input` and fill in the details of the template.

B. Download and use a macro recording tool for your browser. Record the steps to log in and deploy the instance.

C. Use the command `oci compute instance launch --use-last-configuration`

D. Use the `--query` feature

Knowledge Check

Which command would you use to upload a 5GB file to Object Storage using the Command Line Interface as a single part?

A. oci os object get -bn MyBucket --name MyObject --file Myfile.txt

B. oci os object put -bn MyBucket --name MyObject --file Myfile.txt --part-size 512

C. oci os object put -bn MyBucket --name MyObject --file Myfile.txt --no-multipart

D. oci os object upload -bn MyBucket --name MyObject --file Myfile.txt

Knowledge Check

Which command would you use to upload a 5GB file to Object Storage using the Command Line Interface as a single part?

A. oci os object get -bn MyBucket --name MyObject --file Myfile.txt

B. oci os object put -bn MyBucket --name MyObject --file Myfile.txt --part-size 512

C. oci os object put -bn MyBucket --name MyObject --file Myfile.txt --no-multipart

D. oci os object upload -bn MyBucket --name MyObject --file Myfile.txt

Knowledge Check

Your team has recently deployed a custom Java application to a collection of 10 OCI Compute instances. The application is only used 10 hours each day. To save money, you devise a plan to stop the instances at the end of the day when they are no longer needed, and start them each morning shortly before they will be used.

Which tool would provide the easiest method to help implement this plan?

A. Write a custom application using the Java SDK.
B. Use the OCI Command Line Interface.
C. Hire an intern to start and stop resources in the OCI Management Console each day.
D. Use Terraform.

Summary: Command Line Interface

- Reviewed the steps necessary to get the CLI configured
- Looked at some sample code and use cases
- Discussed the benefits of automation with the CLI

Resources CLI Documentation

- Oracle CLI – Getting Started
  https://docs.cloud.oracle.com/iaas/Content/API/Concepts/cliconcepts.htm
- OCI CLI Command Reference
  https://docs.cloud.oracle.com/iaas/tools/oci-cli/latest/oci_cli_docs/oci.html
- JMESpath information
  http://jmespath.org/

ORACLE

ORACLE®
Cloud Infrastructure

cloud.oracle.com/iaas

cloud.oracle.com/tryit