

Cloud for DBAs

Script Your Way Into the Cloud Using Cloud CLI

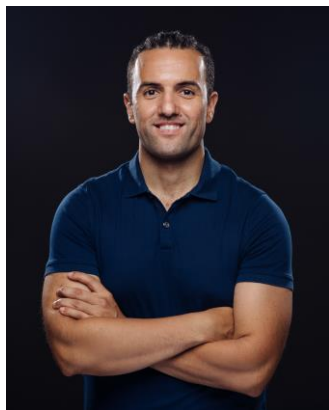
(OCI/Azure/AWS) & Interactive Shell Scripts

Kosseila Hd

Eclipsys Solution Inc

LuxOUG day - Oct 5th





Kousseila Hd

Senior DBA @Eclipsys

-  Blogger
- ~14 years Oracle Experience
- Member of TOUG
- Tech/DevOps enthusiast
- Cloud aficionado
- Beatmaker 😊

 brokedba.blogspot.com

 twitter.com/BrokeDbA

 linkedin.com/in/kousshd

 github.com/brokedba

ORACLE

Certified Expert

Oracle Database 12c:
Oracle RAC and Oracle
Grid Infrastructure
Administrator

ORACLE

Certified Expert

Oracle Database 12c
Data Guard Administrator

ORACLE

Certified Professional

Oracle Database 11g
Administrator

ORACLE

Certified Specialist



Oracle ACE
Pro



AGENDA

1- Cloud Scale Challenges

- ❑ API: “doorway to your Cloud Platform”
- ❑ Automation tools

2- Getting started with the Cloud CLI (OCI):

- ❑ Setup and authentication
- ❑ JSON & JMSPath: “The BFFs”
- ❑ examples

3- Build a stack with my interactive bash scripts

- ❑ GitHub cli projects
- ❑ Difference between OCI/AWS/Azure CLI
- ❑ Demo (network+ website)

4- Q&A

Cloud Scale Challenges

- More resources available
- Flexibility to run and stop services when not needed
- New techs (serverless, containers, K8s)
- Skills debt

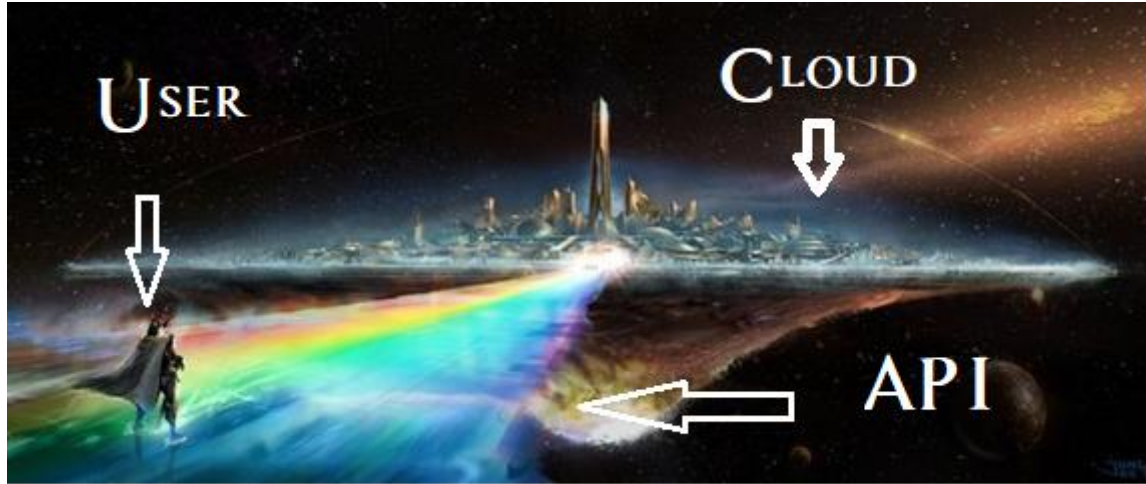
What Automation provides

- Makes it easy to seamlessly adopt and manage new cloud services

Why use Cloud CLI Automation

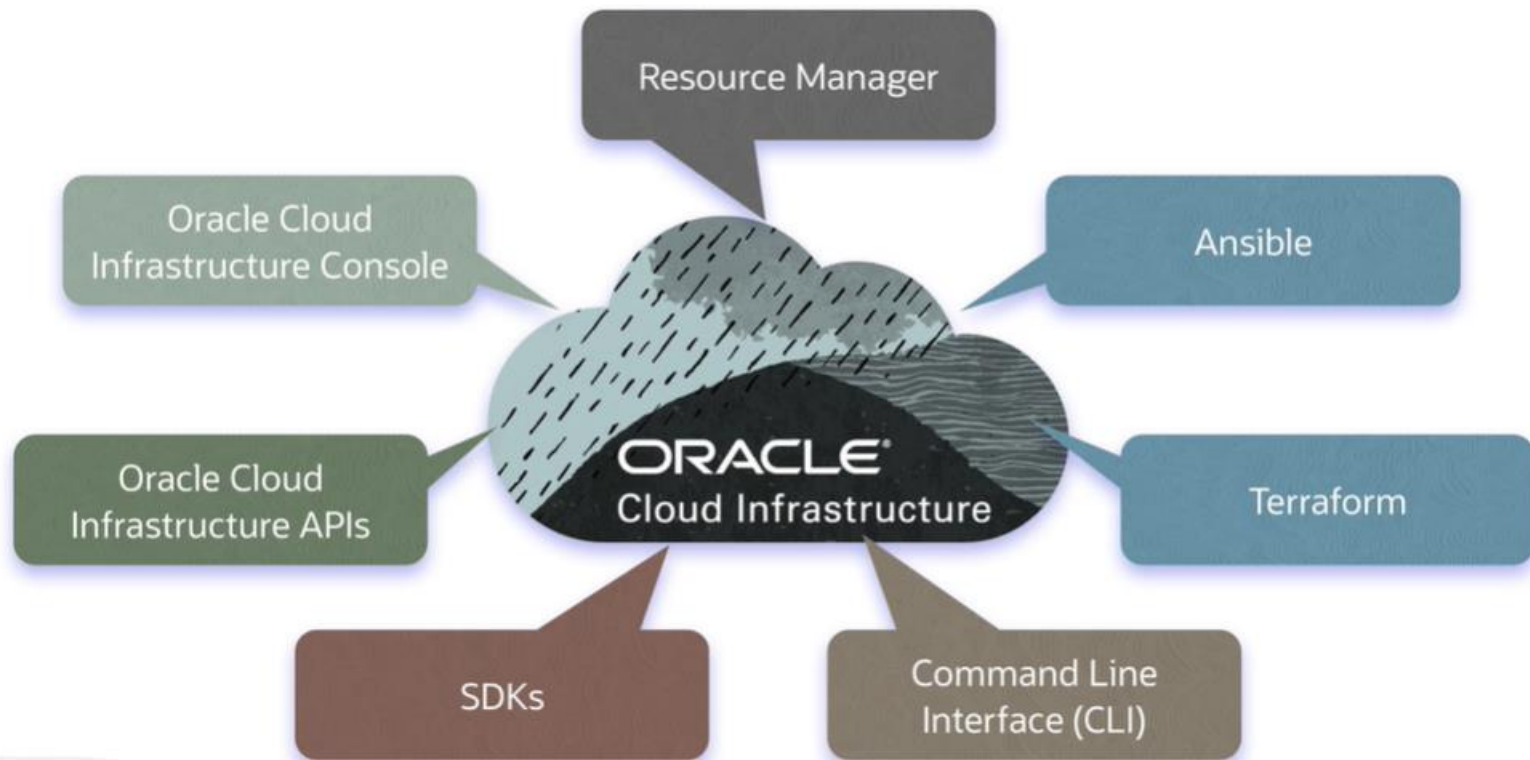
- For executing simple and repeatable tasks (i.e spin batch servers for 2 hrs)
- Most direct and simple way to interact with your cloud infrastructure
- Allow your team to get familiar with the existing/new cloud vendor platform

API: “doorway to your Cloud Platform”



- Rest APIs provided by the Cloud platform is basically the core vehicle that allow anything (infrastructure resource, or cloud services) to be created, deleted or managed.

Different Ways to Access OCI



Automation tools in the cloud

Ansible

Terraform

Command Line
Interface (CLI)

- Configuration management
- Infrastructure as code
- For simple repeatable tasks

Getting started with the Cloud CLI (OCI):

OCI CLI Installation

```
bash -c "$(curl -L https://raw.githubusercontent.com/oracle/oci-cli/master/scripts/install/install.sh)"
```

- Prerequisites
 - Oracle cloud infrastructure account
 - OCI-CLI & python SDK
 - API Signing Key pair (Custom or Autogenerated)
 - Fingerprint

Getting started with the Cloud CLI (OCI):

OCI CLI configuration

```
$ oci setup config
Enter a location for your config [/c/Users/brokedba/.oci/config]: Enter a user OCID:
Enter a tenancy OCID: #
Enter a region : # choose the one defined in your tennacy (Webconsole)
Do you want to generate a new API Signing RSA key pair?
[Y/n]: n
Enter the location of your API Signing private key file:
/c/Users/brokedba/.oci/oci_api_key.pem
Fingerprint: xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Config written to /c/Users/brokedba/.oci/config
```

```
$ cat ~/.oci/config
[DEFAULT]
user=ocid1.user.oc1..aaaaaaaayd2yf6ru5xxxxxxxxxxx
fingerprint=bf:3b:2e:48:a2:98:xx:xx:xx:xx:xx:xx:xx
key_file=C:\Users\brokedba\.oci\oci_api_key.pem
tenancy=ocid1.tenancy.oc1..aaaaaaxxxx
region=ca-toronto-1
```

Getting started with the Cloud CLI (OCI):

Test your first API request

- Create environment variables that stores all your tenancy, user and Compartment ocids

```
export T="ocidl.tenancy.oc1.xxxx"  
export U="ocidl.user.oc1..xxx"  
export C="ocidl.tenancy.oc1..xxx"
```

Features

- Add parameters such as command aliases and predefined queries

```
$ oci setup oci-cli-rc
```

Getting started with the Cloud CLI (OCI):

Test your first API request

```
oci <service> <type> <action> <options>
```

- List default availability domain in the tenancy in JSON

```
$ oci iam availability-domain list
{
  "data": [
    {
      "compartment-id": "ocid1.tenancy.oc1..aaaaaaaxxx",
      "id": "ocid1.availabilitydomain.oc1..aaaaaaaxxxx",
      "name": "BahF:CA-TORONTO-1-AD-1"
    }
  ]
}
```

In Table format

```
oci iam availability-domain list --output table
+-----+-----+-----+
| compartment-id | id | name |
+-----+-----+-----+
| ocid1.tenancy.oc1..aaaa5g4a | ocid1.availabili | BahF:CA-TORONTO-1-AD-1 |
+-----+-----+-----+
```

JSON & JMSPath: “The BFFs”



JSON

Source of truth. Everything that's in cloud is saved in JSON format.

JSON values can be Arrays, objects, or primitives (numbers, strings, Boolean, null).

JMSPATH

```
{"name": "John", "age": 30, "car": null}
```

```
myArray = ["Ford", "BMW", "Fiat"];
```

A query language that allows to narrow the JSON output like SQL and offers:

- Filtering
- Sorting
- Field selection

Test your JMSPath skills



Examples

```
$ oci network vcn list -c $C
{
  "data": [
    {
      "cidr-block": "192.168.0.0/16",
      "compartment-id": "ocidl.tenancy.oc1..ar.....1",
      "default-dhcp-options-id": "ocidl.dhcpoptions.oc1.ca-toronto-1.aaaaaaax3jyje4tpspa7nblmjn3zy6ygd5upr1izpuzr7vk36kx3thqu37a",
      "default-route-table-id": "ocidl.routetable.oc1.ca-toronto-1.aaaaaaac6honjnlpbngxtzz7ksz67jnkum22ffz6gm4bu3btozs6zjb2zq",
      "default-security-list-id": "ocidl.securitylist.oc1.ca-toronto-1.aaaaaaaog3bjtfakpwkpr4lm45r6vhvtkxth56sn7bdo4pwt3v7ap7dt7q",
      "defined-tags": {
        "Oracle-Tags": {
          "CreatedBy": "kosseila@gmail.com",
          "CreatedOn": "2020-08-15T08:49:18.229Z"
        }
      },
      "display-name": "ansi-vcn",
      "dns-label": "ansivcn",
      "freeform-tags": {},
      "id": "ocidl.vcn.oc1.ca-toronto-1.amaaaaaajjjjavaagsxntobyb5poolypd6a2gascmc3cgefhnv2birfr1xra",
      "ipv6-cidr-block": null,
      "ipv6-public-cidr-block": null,
      "lifecycle-state": "AVAILABLE",
      "time-created": "2020-08-15T08:49:18.232000+00:00",
      "vcn-domain-name": "ansivcn.oraclevcn.com"
    }
  ],
}
```

To query the VCN Name, CIDR, Domain and DNS label we can run the below

```
$ oci network vcn list -c $C --output table --query "data[*].{CIDR:\`cidr-block\`, VCN_NAME:\`display-name\`, DOMAIN_NAME:\`vcn-domain-name\`, DNS:\`dns-label\`}"
```




CIDR	DNS	DOMAIN_NAME	VCN_NAME
192.168.0.0/16	ansivcn	ansivcn.oraclevcn.com	ansi-vcn
192.168.64.0/20	terra	terra.oraclevcn.com	Terravcn



Build a stack with CLI and bash scripts

GitHub cli projects



-  [brokedba / oci-cli-examples](#)
-  [brokedba / aws-cli-examples](#)
-  [brokedba / az-cli-examples](#)

Mistakes with IDs and lookups syntax are common,
That's where “**Shell scripts**” come handy.

Related Blog labs: [az-cli](#), [aws-cli](#), [oci-cli](#)



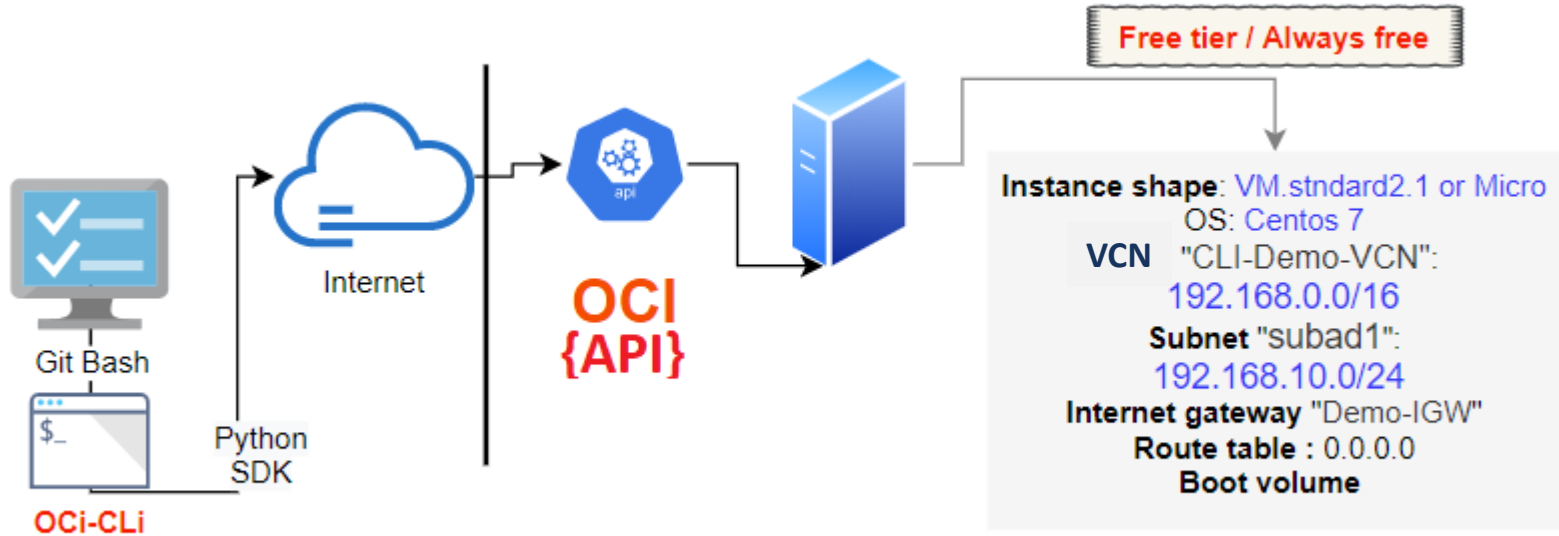
LuxOUG day – Oct 5th



DEMO

OCI

Resources: VCN , SUBNET, Internet gateway, security lists, Compute



Build a stack with CLI and bash scripts

Difference between OCI/AWS/Azure CLI

1. Authentication ([AWS](#)/[AZURE](#)/[GCP](#))
2. Queries and filters (JMSPATH)
3. Cloud-init

Difference between OCI/AWS/Azure CLI

AWS

Run AWS configure

Now that you have installed aws cli along with the access key info gathered in your csv file, you can finally configure your aws-cli with just the key id and the access key (region and output format are not credentials). To do so run the following:

```
$ aws configure
Access Key ID:
AKxxxxxxxxxx
Secret Access Key:
Dxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Default region name [us-east-1]:
Default output format [table]:
```

Difference between OCI/AWS/Azure CLI

AWS

A. **Command structure** : is based on the below components

```
$ aws <AWS service> <operation to perform> [one or more options & parameters]
```

Parameters:

Will be followed by their values, for example when specifying an instance id we want to describe or defining a name for a created key-pair. The value type can also vary (string, integer, JSON, list, binary,...)

Options :

- 1- "-- output" : will format AWS CLI output into Json, yaml, Table, or text (raw).
- 2- "-- query" : Allows to choose the **list of fields** to return in the response. It can be used to do some simple filtering.
- 3- "-- filters" : Is the **condition** used to specify *which resources* you want described or listed.

B. Filters vs Query :

The **--query** option relies on JMSPath and its filtering is done at client side while **--filters** does it at server level which is way faster and more efficient. I personally use **filters** to narrow my research and **query** to specify which field I want to display.

- To demonstrate the nuance, here's an example where we filter an aws region using each option (filters and query)

```
$ aws ec2 describe-regions --query 'Regions[?RegionName==`us-west-2` ]'  
$ aws ec2 describe-regions --filters "Name=region-name,Values=us-west-2"
```

Difference between OCI/AWS/Azure CLI

Azure

A. **Command structure** : is based on the below components

```
$ az [group] [subgroup] [command][parameters]
```

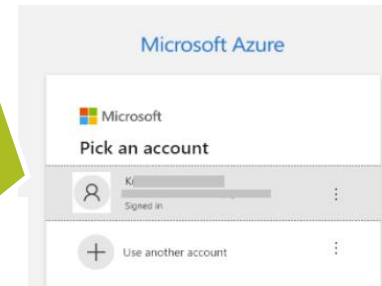
- **Sign in to Azure from AZ CLI**

```
brokedba~$ az login
```

- Once signed in, your web page will confirm that **az cli** is now configured accordingly .

You have logged into Microsoft Azure!

You can close this window, or we will redirect you to the [Azure CLI documents](#) in 10 seconds.



Additional login options are available [here](#)

Difference between OCI/AWS/Azure CLI

Azure

- Account detail

AZ resource group = OCI compartment

```
$ az account list --- OR account show
{
  "environmentName": "AzureCloud",
  "homeTenantId": "6187922f-3371-xxxx-xxxx-xxxxxxxxxxxx",
  "id": "4dc6d66a-f5c9-4f0e-8f01-8b46e413ea93",
  "isDefault": true,
  "managedByTenants": [],
  "name": "BrokeDBA",
  "state": "Enabled",
  "tenantId": "6187922f-3371-xxxx-xxxx-xxxxxxxxxxxx", --> same as homeTenantId
  "user": {
    "name": "xxxxxx@outlook.com",
    "type": "user"
  }
}
```

- Aliases

```
$ az alias create --name rg --command group
$ az alias list
Alias Command
-----
rg      group
```

Help

```
$ az find "az vm"
Finding examples...
Get the details of a VM. (autogenerated)
az vm show --name MyVm --resource-group MyResourceGroup
```

Difference between OCI/AWS/Azure CLI

Azure

- **Cloud init**

The “--custom-data” parameter or cloud-init works only for linux instance not windows

Azure proposed alternative : **RunPowerShellScript** after instance creation

- **ID**: Azure doesn't provide regular alpha numeric ids for its resources but a sort of path-based identification

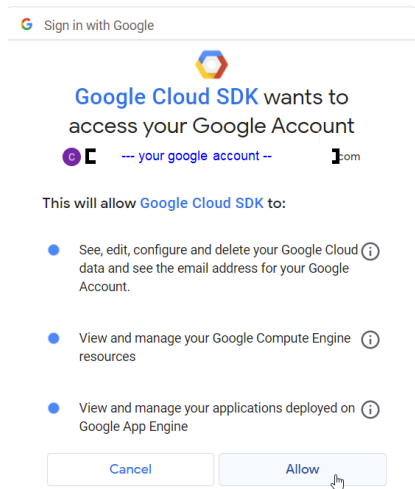
```
$ SUBNET_ID  
/subscriptions/xx/resourceGroups/my_group/providers/Microsoft.Network/virtualNetworks/MY-  
VNET/subnets/My_SUBNET
```

Difference between OCI/AWS/Azure CLI

Oops! Did you say GCP?

All you need is run `gcloud init` command to:


1. Authorize Cloud SDK to access the GCP platform using your user account
2. Set configuration parameters like current project or default GCE region/zone etc..



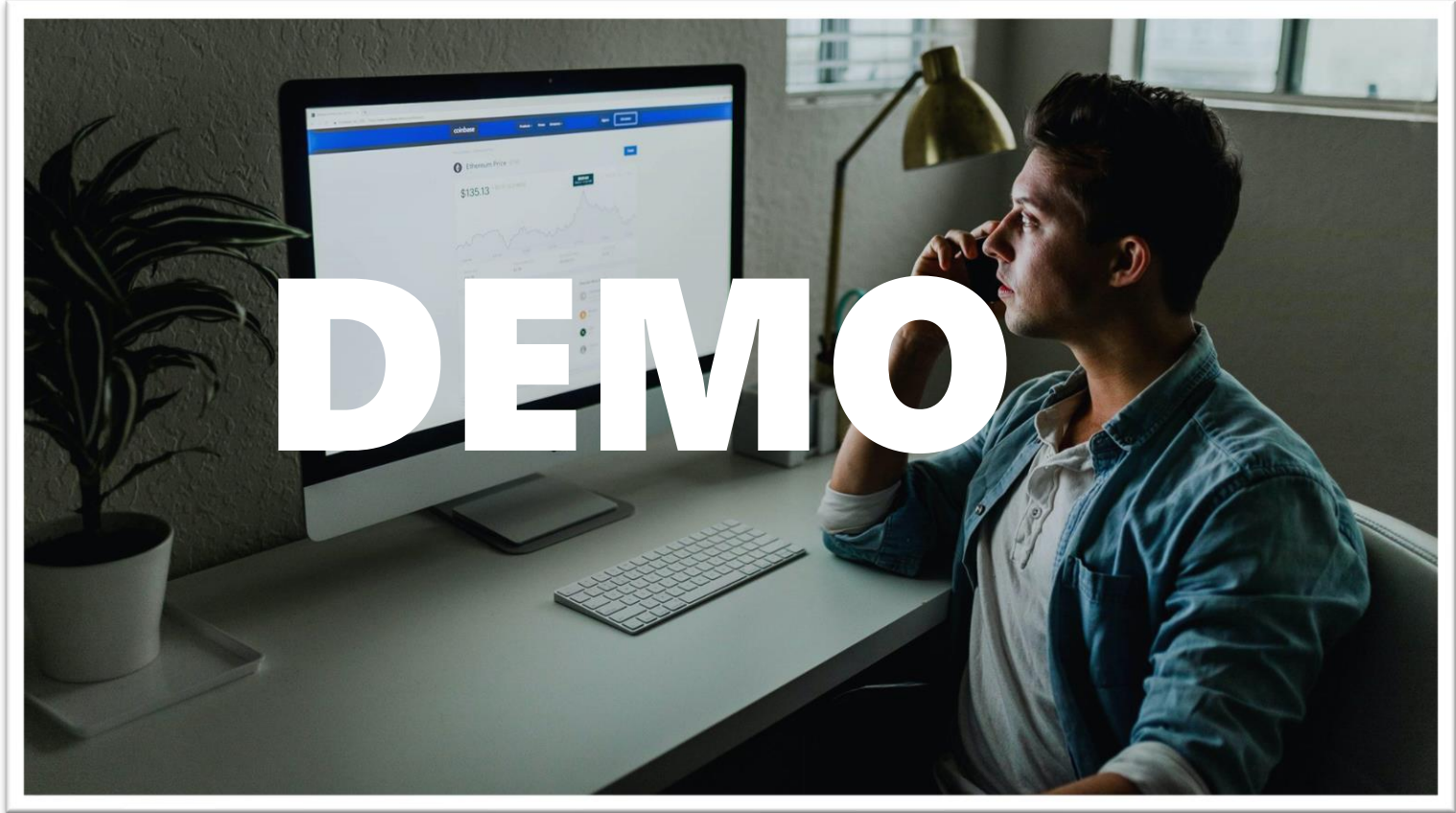
Google

Sign in

Please copy this code, switch to your application and paste it there:

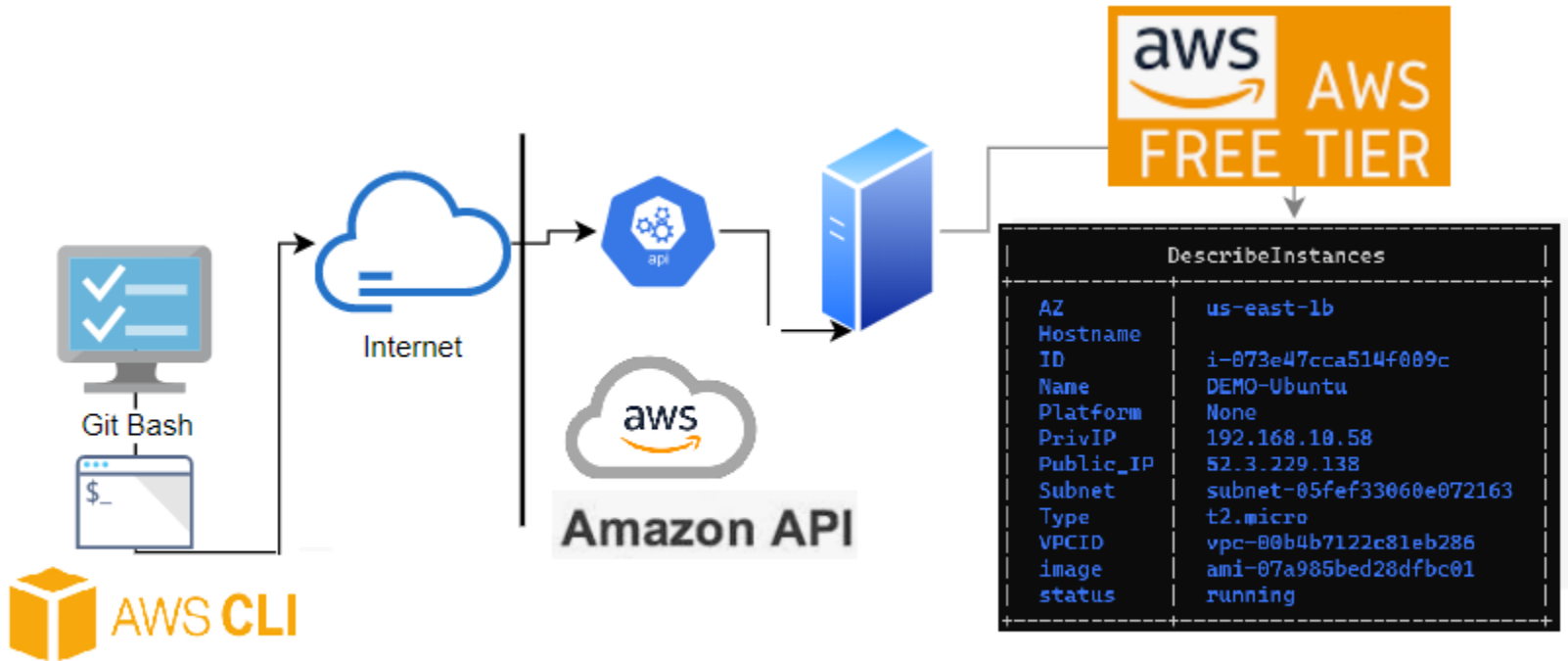
`kN5wabUBHqd2SXcwMV4/1AX4XfwIzu7m9uq1RkzBPi_jMIDBw3toc10c4srzwc` 

Deploy using Cloud CLI



DEMO

Resources: VPC , SUBNET, Internet gateway, route, security group, Web Compute
AWS



DEMO

AWS

- Below are the files you will retrieve after cloning the directory (including the description)

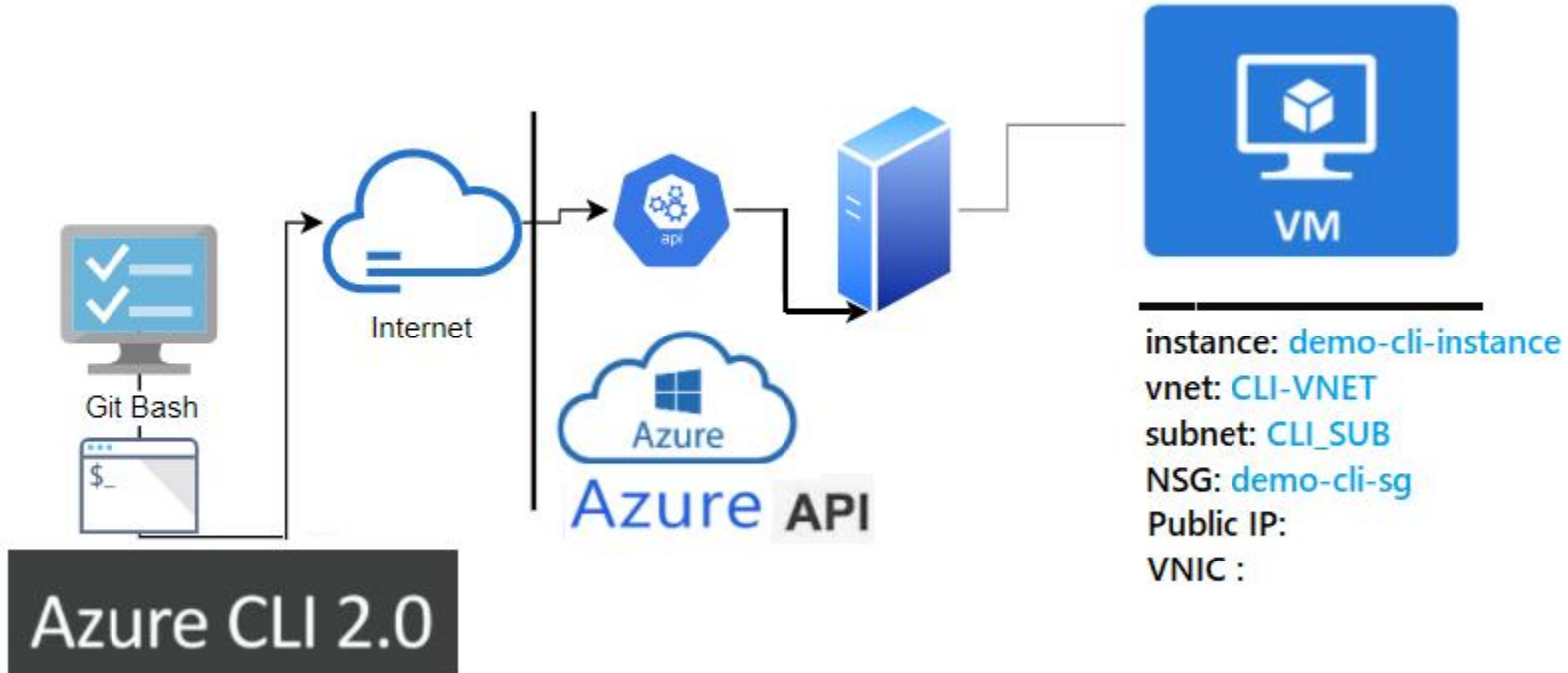
```
$ cd ./aws-cli-examples
$ tree
|-- check_image.sh      ---> Displays most recent AMI details per OS type
|-- create_igateway.sh ---> Create an Internet Gateway to link to a route table
|-- create_instance.sh ---> Launch an instance (requires a vpc and subnet)
|-- create_route.sh    ---> Create a route table & add a route to internet
|-- create_subnet.sh   ---> Create a subnet within the VPC
|-- create_vpc.sh      ---> Create a VPC with dedicated security group
|-- cloud-init         ---> Subfolder containing bootstrap code for each OS
  |-- amzl_userdata.txt ---> userdata script for amazon linux 2 instance
  |-- el_userdata.txt   ---> userdata script for CENTOS and RHEL7 instance
  |-- sles_userdata.txt ---> userdata script for SUSE 15 instance
  |-- ubto_userdata.txt ---> userdata script for Ubuntu instance
  |-- win_userdata.txt  ---> userdata script for Windows server instance
```

[aws-cli](#)

DEMO

Azure

Resources: VNET, SUBNET, PublicIP, VNIC, NSG, Web Compute



DEMO

Azure

- Below are the files you will find after cloning the directory (with embedded hyperlinks)

```
$ cd ./az-cli-examples
$ tree
|-- check\_az\_image.sh      ---> Displays most recent image details per OS type
|-- check\_az\_vmsize.sh     ---> Displays available vmsize per chosen vcpu number
|-- create\_az\_instance.sh  ---> Launch a vm (requires a vnet)
|-- create\_az\_rg.sh        ---> Create a resource group
|-- create\_az\_subnet.sh    ---> Create a subnet within the VNET
|-- create\_az\_vnet.sh      ---> Create a VNET+subnet with its security group
|-- cloud-init              ---> Subfolder containing bootstrap code for each OS
  |-- centos\_userdata.txt   ---> userdata script for CENTOS 7
  |-- olinux\_userdata.txt  ---> userdata script for Oracle linux 7
  |-- rhel\_userdata.txt    ---> userdata script for RHEL7
  |-- sles\_userdata.txt   ---> userdata script for SUSE 15
  |-- ubto\_userdata.txt   ---> userdata script for Ubuntu
  |-- win\_userdata.ps1    ---> PowerShell script for Windows server 2016
```

Script Your Way Into the Cloud

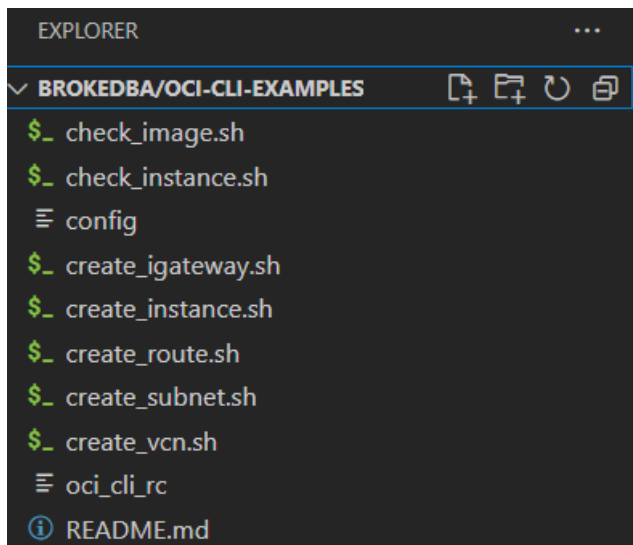
What's NEXT

- Clone/fork/Improve the scripts repo and Try it yourself
- Automate other tasks: stop all vms/db instances/patch/upgrade DB
- Start your own version for other resources
 - Network resources : load balancers , public Ips, VPN
 - Compute: Kubernetes, instance groups, Batch servers
 - Database services
 - storage
- Explore GCP equivalents
- Stay Curious

DEMO

Github to Visual Code Tip

<https://github.com/brokedba/oci-cli-examples> turns your repo into VC view



Cloud for DBAs



LuxOUG day - Oct 5th

Cloud for DBAs

Kosseila Hd
Senior DBA @Eclipsys



GitHub



brokedba.blogspot.com



twitter.com/BrokeDbA



linkedin.com/in/kousshd

<https://github.com/brokedba>

LuxOUG day – Oct 5th

