Step-by-step Guide to Deploy Applications to Akash Cloud Network

Akash is an open-source decentralized cloud network, where you can host ***any*** application currently running on the cloud. The deployment is off-chain and performed over a private peer-to-peer network isolated from the blockchain, and asset transfer occurs off-chain to protect and performance that we need for a mission critical application running on the Cloud. Unlike centralized providers, in Akash, *the users (tenants) set the price and the terms, the provider bids for their workloads*.

MY APPLICATION LINK: 5rc5abldqpdvnaju1l1rc3egug.ingress.ewr1p0.mainnet.akashian.io

PREREQUISITE (I RECOMMEND YOU TO USE LINUX OR MACOS MACHINE):

1) **INSTALL DOCKER ON YOUR MACHINE**
2) **MAKE AN ACCOUNT ON DOCKER HUB**
3) **INSTALL "GO" ON YOUR MACHINE**
4) **FINALLY INSTALL AKASH CLI ON YOUR MACHINE**

CONTAINERIZING YOUR APPLICATION:

For demonstration purposes, we'll use this web application.

Clone the application by:

```
git clone https://github.com/Developer-piyush/AKASHCLOUD
cd AKASHCLOUD/
```

TO DEPLOY ON AKASH, WE NEED TO CONTANERIZE OUR APPLICATION WITH DOCKER:

Inside the `AKASHCLOUD` directory, create a file called `Dockerfile` with the following content:

```
FROM nginx:alpine
COPY . /usr/share/nginx/html
```

Build the Docker image (we'll name it `webserver-image` and tag it as `v1` for simplicity;) via:

```
docker build -t webserver-image:v1 .
```

To test it, run the image as a container, with the host's port 80 forwarded to the container's port 80 (default port of Nginx):

```
docker run -p 80:80 webserver-image:v1
```

you should be able to view the application at `http://localhost:80`.

PUSH IMAGE TO DOCKER REGISTRY:

Once we have a built Docker image, we need to make it publicly available so that the Akash deployment process can pull the image. If you have a Docker Hub account with username `USERNAME (IN MY CASE IT IS Piyushch)`, create a repository called `webserver-image`, and then tag and push the image with:

```
docker tag webserver-image:v1 piyushch/webserver-image:v1
docker push piyushch/webserver-image:v1
```

NOTE: in place of Piyushch and webserver-image:v1 (you should use your own docker hub username and docker image name)

we're done! The Docker image representing our containerized application is now publicly available, and we can move on to deployment to Akash.

DEPLOYING TO AKASH:

After writing and containerizing your application, deploying to Akash simply involves writing small configuration file and executing a couple of commands.

ENVIRONMENT SETUP TO DEPLOY:

OPEN YOUR TERMINAL IN ROOT DIRECTORY (IN OUT CASE IT IS AKASHCLOUD) AND RUN FOLLOWING COMMANDS:

```
AKASH_NET="https://raw.githubusercontent.com/ovrclk/net/master/mainnet"
```

```
AKASH_VERSION="$(curl -s "$AKASH_NET/version.txt")"
```

```
export AKASH_CHAIN_ID="$(curl -s "$AKASH_NET/chain-id.txt")"
```

```
export AKASH_NODE="$(curl -s "$AKASH_NET/rpc-nodes.txt" | shuf -n 1)"

echo $AKASH_NODE
```

```
curl -s "$AKASH_NET/api-nodes.txt" | shuf -n 1
```

```
AKASH_KEY_NAME=Piyushch      (NAME OF YOUR CHOICE IN OUR CASE IT IS PIYUSHCH)
```

```
AKASH_KEYRING_BACKEND=os
```

```
akash \
  --keyring-backend "$AKASH_KEYRING_BACKEND" \
  keys add "$AKASH_KEY_NAME"
```

(above command will generate new akash wallet for you, save account address and mnemonic key at safe place)

```
akash \
  --keyring-backend "$AKASH_KEYRING_BACKEND" \
  keys export "$AKASH_KEY_NAME"
```

(Above command will export your private keys, after running this command it will ask for passphrase(password) to decrypt your keys: simply type the password of your choice and remember this password or save it at safe place)

```
akash \
  --keyring-backend "$AKASH_KEYRING_BACKEND" \
  keys show "$AKASH_KEY_NAME" -a
```

(Above command will display your account address which you have generated)

```
AKASH_ACCOUNT_ADDRESS=<YOUR ACCOUNT ADDRESS>
```

NOW SEND/ADD SOME FUNDS TO THE WALLET FOR DEPLOYING OUR APP ON MAINNET:

Tokens may currently be purchased on exchanges listed here. From there you can withdraw tokens to your address.

```
akash \
 --node "$AKASH_NODE" \
 query bank balances "$AKASH_ACCOUNT_ADDRESS"
```

(ABOVE COMMAND WILL DISPLAY YOUR ACCOUNT BALANCE)

Create The Deployment Configuration:

Create a deployment configuration deploy.yml in your root directory to deploy

Piyushch/webserver-image:v1

ADD FOLLOWING CONTENT TO deploy.yml file:

```
---
version: "2.0"

services:
  web:
    image: Piyushch/webserver-image:v1
    expose:
      - port: 80
        as: 80
        to:
          - global: true

profiles:
  compute:
    web:
      resources:
        cpu:
          units: 0.1
        memory:
          size: 512Mi
        storage:
          size: 512Mi
  placement:
    westcoast:
      attributes:
        host: akash
      signedBy:
        anyOf:
          - "akash1365yvmc4s7awdyj3n2sav7xfx76adc6dnmlx63"
      pricing:
        web:
          denom: uakt
          amount: 1000

deployment:
  web:
    westcoast:
      profile: web
      count: 1
```

JUST CHANGE DOCKER IMAGE LINK IN ABOVE CODE:

image: **USERNAME/DOCKERIMAGE**

IN OUT CASE IT IS: **Piyushch/webserver-image:v1**

CREATE THE DEPLOYMENT:

To create a deployment, a certificate must first be created. To do this, Just run the following command, it will ask for transaction (type Y and enter):

```
akash tx cert create client --chain-id $AKASH_CHAIN_ID --keyring-backend $AKASH_KEYRING_BACKEND --from $AKASH_KEY_NAME --node $AKASH_NODE --fees 5000uakt
```

to deploy on akash run:

```
akash tx deployment create deploy.yml --from $AKASH_KEY_NAME --node $AKASH_NODE --chain-id $AKASH_CHAIN_ID --fees 5000uakt -y
```

Above command will generate response similar to:


```
{
 "height":"140325",
 "txhash":"2AF4A01B9C3DE12CC4094A95E9D0474875DFE24FD088BB443238AC06E36D98EA",
 "codespace":"",
 "code":0,
 "data":"0A130A116372656174652D6465706C6F796D656E74",

"raw_log":"[{\"events\":[{\"type\":\"akash.v1\",\"attributes\":[{\"key\":\"module\",\"value\":\"deployment\"},{\"key\":\"action\",\"value\":\"deployment-created\"},{\"key\":\"version\",\"value\":\"2b86f778de8cc9df415490efa162c58e7a0c297fbac9cdb8d6c6600eda56f17e\"},{\"key\":\"owner\",\"value\":\"akash1vn06ycjjnvsvl639fet9lajjctuturrtx7fvuj\"},{\"key\":\"dseq\",\"value\":\"140324\"},{\"key\":\"module\",\"value\":\"market\"},{\"key\":\"action\",\"value\":\"order-created\"},{\"key\":\"owner\",\"value\":\"akash1vn06ycjjnvsvl639fet9lajjctuturrtx7fvuj\"},{\"key\":\"dseq\",\"value\":\"140324\"},{\"key\":\"gseq\",\"value\":\"1\"},{\"key\":\"oseq\",\"value\":\"1\"}]},{\"type\":\"message\",\"attributes\":[{\"key\":\"action\",\"value\":\"create-deployment\"},{\"key\":\"sender\",\"value\":\"akash1vn06ycjjnvsvl639fet9lajjctuturrtx7fvuj\"},{\"key\":\"sender\",\"value\":\"akash1vn06ycjjnvsvl639fet9lajjctuturrtx7fvuj\"}]},{\"type\":\"transfer\",\"attributes\":[{\"key\":\"recipient\",\"value\":\"akash17xpfvakm2amg962yls6f84z3kell8c5lazw8j8\"},{\"key\":\"sender\",\"value\":\"akash1vn06ycjjnvsvl639fet9lajjctuturrtx7fvuj\"},{\"key\":\"amount\",\"value\":\"5000uakt\"},{\"key\":\"recipient\",\"value\":\"akash14pphss726thpwws3yc458hggufynm9x77l4l2u\"},{\"k
```

ey\":\"sender\",\"value\":\"akash1vn06ycjjnvsvl639fet9lajjctuturrtx7fvuj\"},{\"key\":\"amount\",\"value\":\"5000000uakt\"}]}]}]",
"logs":[
{
"msg_index":0,
"log":"",
"events":[
{
"type":"akash.v1",
"attributes":[
{
"key":"module",
"value":"deployment"
},
{
"key":"action",
"value":"deployment-created"
},
{
"key":"version",
"value":"2b86f778de8cc9df415490efa162c58e7a0c297fbac9cdb8d6c6600eda56f17e"
},
{
"key":"owner",
"value":"akash1vn06ycjjnvsvl639fet9lajjctuturrtx7fvuj"
},
{
"key":"dseq",
"value":"140324"
},
{
"key":"module",
"value":"market"
},
{
"key":"action",
"value":"order-created"
},
{
"key":"owner",
"value":"akash1vn06ycjjnvsvl639fet9lajjctuturrtx7fvuj"
},
{
"key":"dseq",
"value":"140324"
},
{
"key":"gseq",

```
      "value":"1"
     },
     {
      "key":"oseq",
      "value":"1"
     }
    ]
   },
   {
    "type":"message",
    "attributes":[
     {
      "key":"action",
      "value":"create-deployment"
     },
     {
      "key":"sender",
      "value":"akash1vn06ycjjnvsvl639fet9lajjctuturrtx7fvuj"
     },
     {
      "key":"sender",
      "value":"akash1vn06ycjjnvsvl639fet9lajjctuturrtx7fvuj"
     }
    ]
   },
   {
    "type":"transfer",
    "attributes":[
     {
      "key":"recipient",
      "value":"akash17xpfvakm2amg962yls6f84z3kell8c5lazw8j8"
     },
     {
      "key":"sender",
      "value":"akash1vn06ycjjnvsvl639fet9lajjctuturrtx7fvuj"
     },
     {
      "key":"amount",
      "value":"5000uakt"
     },
     {
      "key":"recipient",
      "value":"akash14pphss726thpwws3yc458hggufynm9x77l4l2u"
     },
     {
      "key":"sender",
      "value":"akash1vn06ycjjnvsvl639fet9lajjctuturrtx7fvuj"
     },
```

```
        {
          "key":"amount",
          "value":"5000000uakt"
        }
       ]
      }
    ]
   }
 ],
 "info":"",
 "gas_wanted":"200000",
 "gas_used":"94653",
 "tx":null,
 "timestamp":""
}
```


FROM ABOVE RESULT WE NEED VALUE OF dseq, gseq and oseq (you can see the bold part in above result, we have dseq=140324, gseq=1 and oseq=1):

Now simple run following commands to set shell variables dseq, gseq and oseq:

```
AKASH_DSEQ=140324
AKASH_GSEQ=1
AKASH_OSEQ=1

echo $AKASH_DSEQ $AKASH_OSEQ $AKASH_GSEQ
```

TO VERIFY DEPLOYMENT RUN:

```
akash query deployment get --owner $AKASH_ACCOUNT_ADDRESS --node $AKASH_NODE --dseq $AKASH_DSEQ
```

TO VERIFY ORDER CREATION RUN:

```
akash query market order get --node $AKASH_NODE --owner $AKASH_ACCOUNT_ADDRESS --dseq $AKASH_DSEQ --oseq $AKASH_OSEQ --gseq $AKASH_GSEQ
```

TO VIEW YOUR BIDS RUN:

After a short time, you should see bids from providers for this deployment with the following command:

```
akash query market bid list --owner=$AKASH_ACCOUNT_ADDRESS --node $AKASH_NODE --dseq $AKASH_DSEQ
```

ABOVE COMMAND WILL GENERATE FOLLOWING RESULT:

```
bids:
- bid:
    bid_id:
      dseq: "140324"
      gseq: 1
      oseq: 1
      owner: akash1vn06ycjjnvsvl639fet9lajjctuturrtx7fvuj
      provider: akash10cl5rm0cqnpj45knzakpa4cnvn5amzwp4lhcal
    created_at: "140326"
    price:
      amount: "1"
      denom: uakt
    state: open
  escrow_account:
    balance:
      amount: "50000000"
      denom: uakt
    id:
      scope: bid
      xid:
akash1vn06ycjjnvsvl639fet9lajjctuturrtx7fvuj/140324/1/1/akash10cl5rm0cqnpj45knzakpa4cnvn5amzwp4lhcal
    owner: akash10cl5rm0cqnpj45knzakpa4cnvn5amzwp4lhcal
    settled_at: "140326"
    state: open
    transferred:
      amount: "0"
      denom: uakt
- bid:
    bid_id:
      dseq: "140324"
      gseq: 1
      oseq: 1
      owner: akash1vn06ycjjnvsvl639fet9lajjctuturrtx7fvuj
      provider: akash1f6gmtjpx4r8qda9nxjwq26fp5mcjyqmaq5m6j7
    created_at: "140326"
    price:
      amount: "1"
```

```
    denom: uakt
  state: open
 escrow_account:
  balance:
   amount: "50000000"
   denom: uakt
  id:
   scope: bid
   xid:
akash1vn06ycjjnvsvl639fet9lajjctuturrtx7fvuj/140324/1/1/akash1f6gmtjpx4r8qda9nxjwq26fp5mcjyqma
q5m6j7
  owner: akash1f6gmtjpx4r8qda9nxjwq26fp5mcjyqmaq5m6j7
  settled_at: "140326"
  state: open
  transferred:
   amount: "0"
   denom: uakt
```

YOU CAN SEE WE GOT BID FROM TWO PROVIDERS BOTH WERE ASKING FOR A PRICE OF

1uakt = [(no of uakt) x (10^(-6))] AKT**:**

WE WILL SELECT SECOND PROVIDER THAT IS:

provider: akash1f6gmtjpx4r8qda9nxjwq26fp5mcjyqmaq5m6j7

NOW SET PROVIDER SHELL VARIABLE BY RUNNING FOLLOWING COMMAND:

```
AKASH_PROVIDER= akash1f6gmtjpx4r8qda9nxjwq26fp5mcjyqmaq5m6j7

echo $AKASH_PROVIDER
```

CREATE YOUR LEASE BY RUNNING FOLLOWING COMMAND:

```
akash tx market lease create --chain-id $AKASH_CHAIN_ID --node $AKASH_NODE --owner
$AKASH_ACCOUNT_ADDRESS --dseq $AKASH_DSEQ --gseq $AKASH_GSEQ --oseq $AKASH_OSEQ --
provider $AKASH_PROVIDER --from $AKASH_KEY_NAME --fees 5000uakt
```

(ABOVE COMMAND WILL ASK FOR TRANSACTION, TYPE Y AND ENTER)

Wait for your Lease (RUN FOLLOWING COMMAND):

```
akash query market lease list --owner $AKASH_ACCOUNT_ADDRESS --node $AKASH_NODE --dseq $AKASH_DSEQ
```

Upload Manifest (RUN FOLLOWING COMMAND):

```
akash provider send-manifest deploy.yml --node $AKASH_NODE --dseq $AKASH_DSEQ --provider $AKASH_PROVIDER --home ~/.akash --from $AKASH_KEY_NAME
```

(You should expect no output from the above command.)

Now that the manifest is uploaded, your image is deployed. You can retrieve the access details by running the below:

```
akash provider lease-status --node $AKASH_NODE --home ~/.akash --dseq $AKASH_DSEQ --from $AKASH_KEY_NAME --provider $AKASH_PROVIDER
```

ABOVE COMMAND WILL GIVE RESPONSE SIMILAR TO:


```
{
 "services": {
  "web": {
   "name": "web",
   "available": 1,
   "total": 1,
   "uris": [
    "rga3h05jetf9h3p6dbk62m19ck.ingress.ewr1p0.mainnet.akashian.io"
   ],
   "observed_generation": 1,
   "replicas": 1,
   "updated_replicas": 1,
   "ready_replicas": 1,
   "available_replicas": 1
  }
 },
 "forwarded_ports": {}
}
```


YOU CAN ACCESS YOUR APPLICATION AT:

rga3h05jetf9h3p6dbk62m19ck.ingress.ewr1p0.mainnet.akashian.io

I HAVE SUCCESSFULLY DEPLOYED MY APPLICATION ON AKASH CLOUD JUST FOLLOWING ABOVE STEPS:

LINK: 5rc5abldqpdvnaju1l1rc3equg.ingress.ewr1p0.mainnet.akashian.io

FOR MORE GUIDANCE YOUR CAN FOLLOW AKASH DOC FOR BEGINNERS:

(CLICK HERE)