TINKERBELL

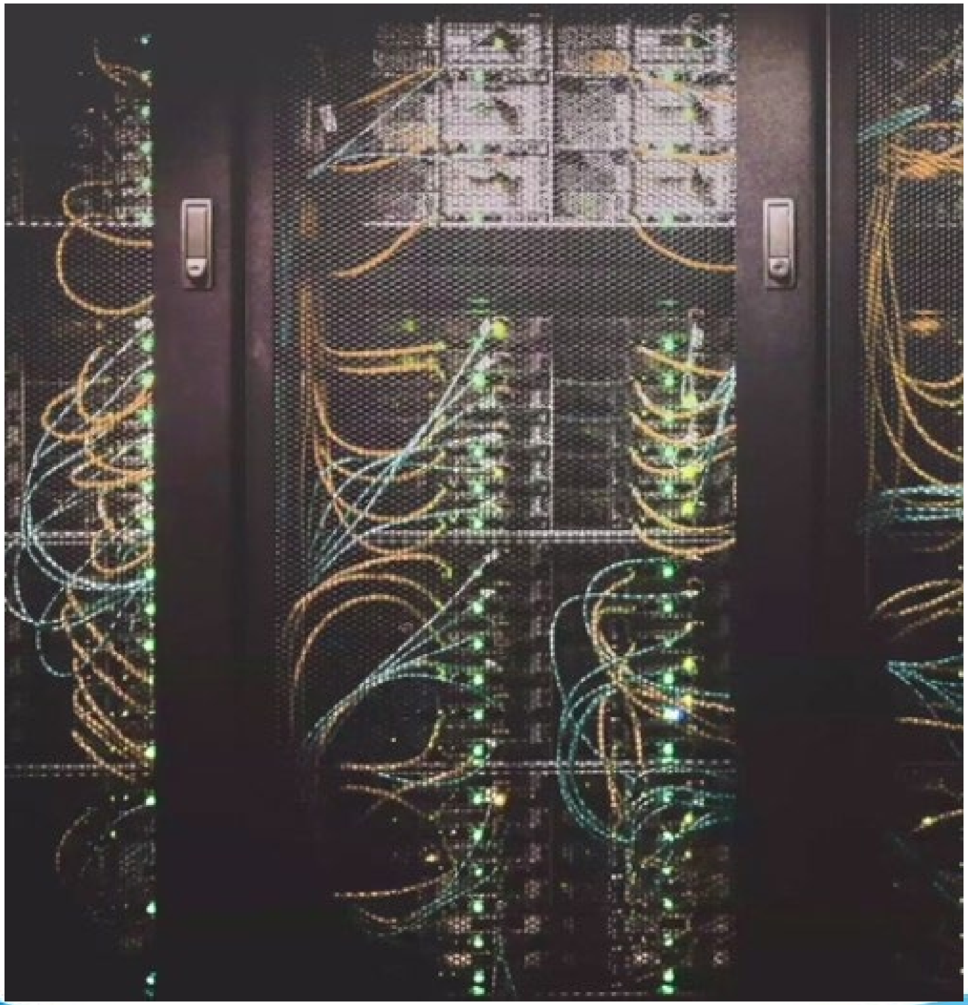
An Automated Bare Metal Provisioning Engine

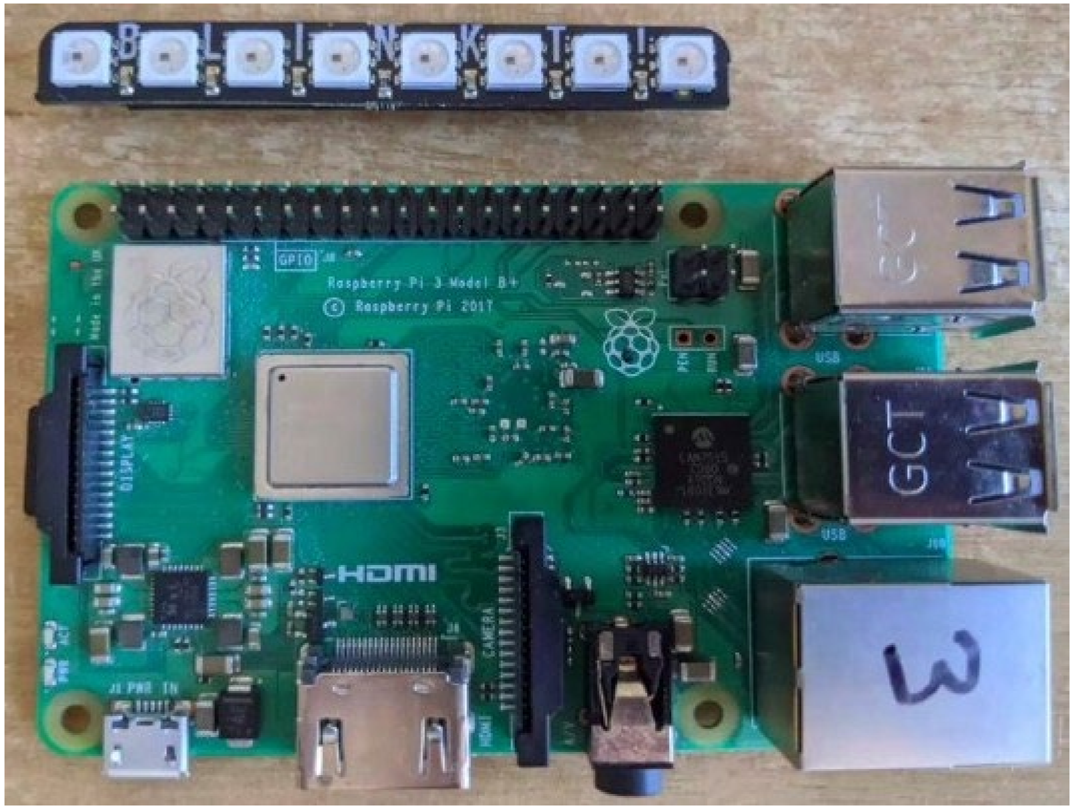
By:
Aman Parauliya
Senior Software Engineer at InfraCloud Technologies

Agenda

- **What is “Bare Metal” ?**
  - Servers
  - Network Booting
  - Use Cases
  - Challanges
- **Tinkerbell : A Complete Solution**
  - Components and their usage
  - Tink – the workflow engine
  - Architecture of Tink
    - Hardware Inventory
    - Template – A Yaml based definition
    - Workflow : Template + Targeted Hardware
- **Demo**

What is Bare Metal?





Bare Metal - Servers

- Network - (Support of IPMI)
- Storage
- Boot Environment - (Support of iPXE)

Bare Metal – Network Booting

- PXE/iPXE
- DHCP – Providing IP dynamically
- TFTP – Provides Initial FileSystem
- NFS – If you don’t have the storage in your hardware

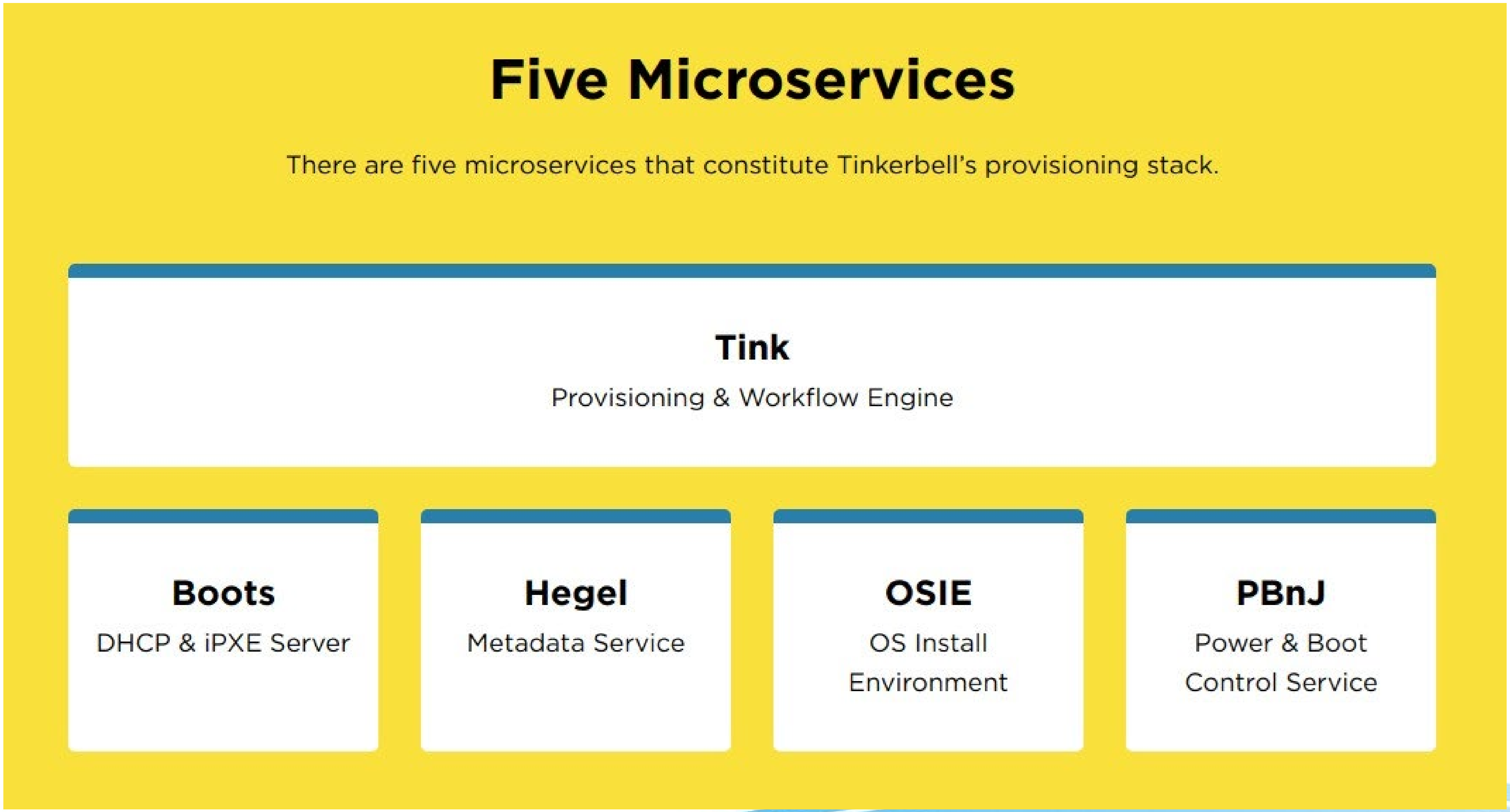
Bare Metal - Use cases

- Existing Infrastructure Services
- Data Security
- Latency
- Consistant and Predictable Performance

Bare Metal - Challanges

- Difficult to manage the large infrastrcture
- Different CPU like Intel, ARM, different distros
- Increase of Control comes with Increase of complexity

TINKERBELL : Components



Five Microservices

There are five microservices that constitute Tinkerbell's provisioning stack.

Tink
Provisioning & Workflow Engine

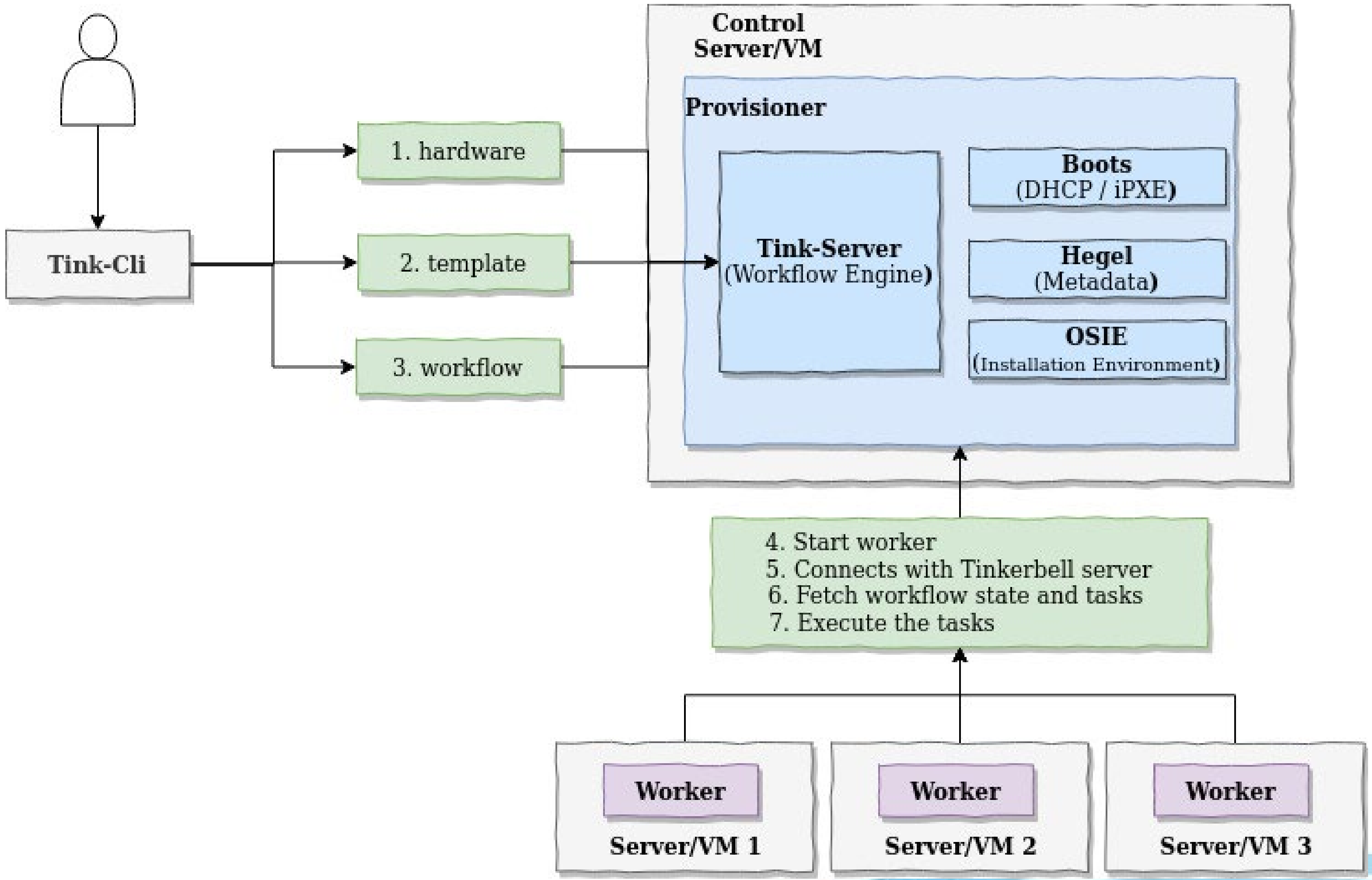
Boots
DHCP & iPXE Server

Hegel
Metadata Service

OSIE
OS Install Environment

PBnJ
Power & Boot Control Service

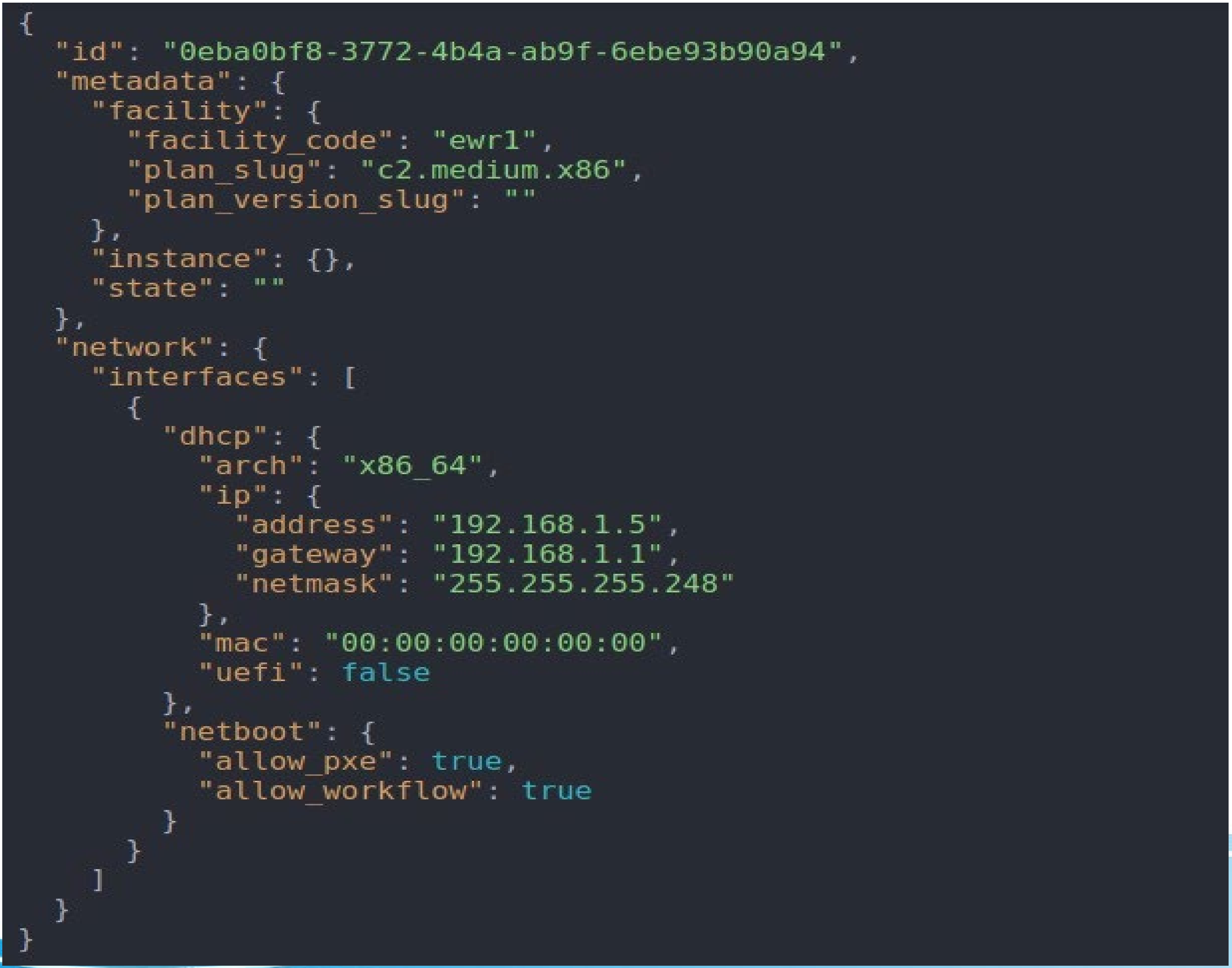
Tink – A workflow Engine



TINKERBELL : Control plane - Provisioner

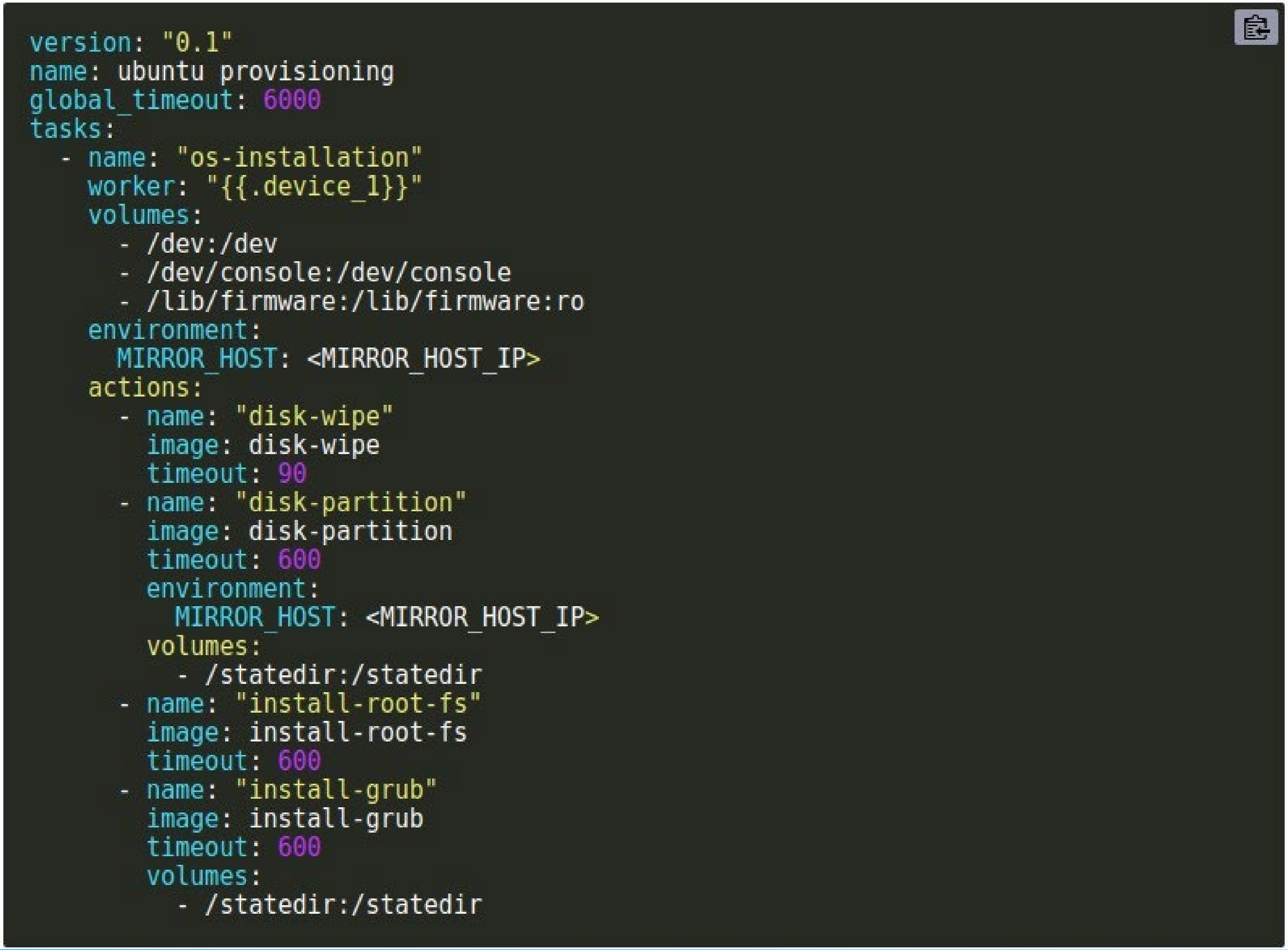
- Provisioner is a Network server on which all the services of tinkerbell are running in docker containers:
  - **Boots** – DHCP and TFTP
  - **Hegel** – Provide Metadata of a machine
  - **Tink-server** – Workflow Engine
  - **Tink-cli** – Client to interact with Tink-server
  - **Postgres** – DB to store data/metadata/events
  - **Registry** – Private docker registry in which imgae of **Tink-worker**
  - and actions will be stored

Hardware Data



```
{
  "id": "0eba0bf8-3772-4b4a-ab9f-6ebe93b90a94",
  "metadata": {
    "facility": {
      "facility_code": "ewr1",
      "plan_slug": "c2.medium.x86",
      "plan_version_slug": ""
    },
    "instance": {},
    "state": ""
  },
  "network": {
    "interfaces": [
      {
        "dhcp": {
          "arch": "x86_64",
          "ip": {
            "address": "192.168.1.5",
            "gateway": "192.168.1.1",
            "netmask": "255.255.255.248"
          },
          "mac": "00:00:00:00:00:00",
          "uefi": false
        },
        "netboot": {
          "allow_pxe": true,
          "allow_workflow": true
        }
      }
    ]
  }
}
```

Workflow Definition: A YAML based Template



```yaml
version: "0.1"
name: ubuntu provisioning
global_timeout: 6000
tasks:
  - name: "os-installation"
    worker: "{{.device_1}}"
    volumes:
      - /dev:/dev
      - /dev/console:/dev/console
      - /lib/firmware:/lib/firmware:ro
    environment:
      MIRROR_HOST: <MIRROR_HOST_IP>
    actions:
      - name: "disk-wipe"
        image: disk-wipe
        timeout: 90
      - name: "disk-partition"
        image: disk-partition
        timeout: 600
        environment:
          MIRROR_HOST: <MIRROR_HOST_IP>
        volumes:
          - /statedir:/statedir
      - name: "install-root-fs"
        image: install-root-fs
        timeout: 600
      - name: "install-grub"
        image: install-grub
        timeout: 600
        volumes:
          - /statedir:/statedir
```

Create Workflow : With Tink-CLI

1. Push the hardware data of the worker machine in the db

```
docker exec deploy_tink-cli_1 \
  tink hardware push --file data.json
```

2. Create the template to define a workflow

```
docker exec deploy_tink-cli_1 \
  tink template create -n <template name>(Unique) -p
<path to template file>
```

3. Create the workflow

```
docker exec deploy_tink-cli_1 \
  tink workflow create -t <template uuid> -r ‘{“device_1”:
“MAC/IP address”}’
```

DEMO

Links:

1. https://tinkerbell.org/
2. https://github.com/tinkerbell/

Few Community Works

- https://github.com/alexellis/tinkerbot
- https://github.com/tinkerbell/portal