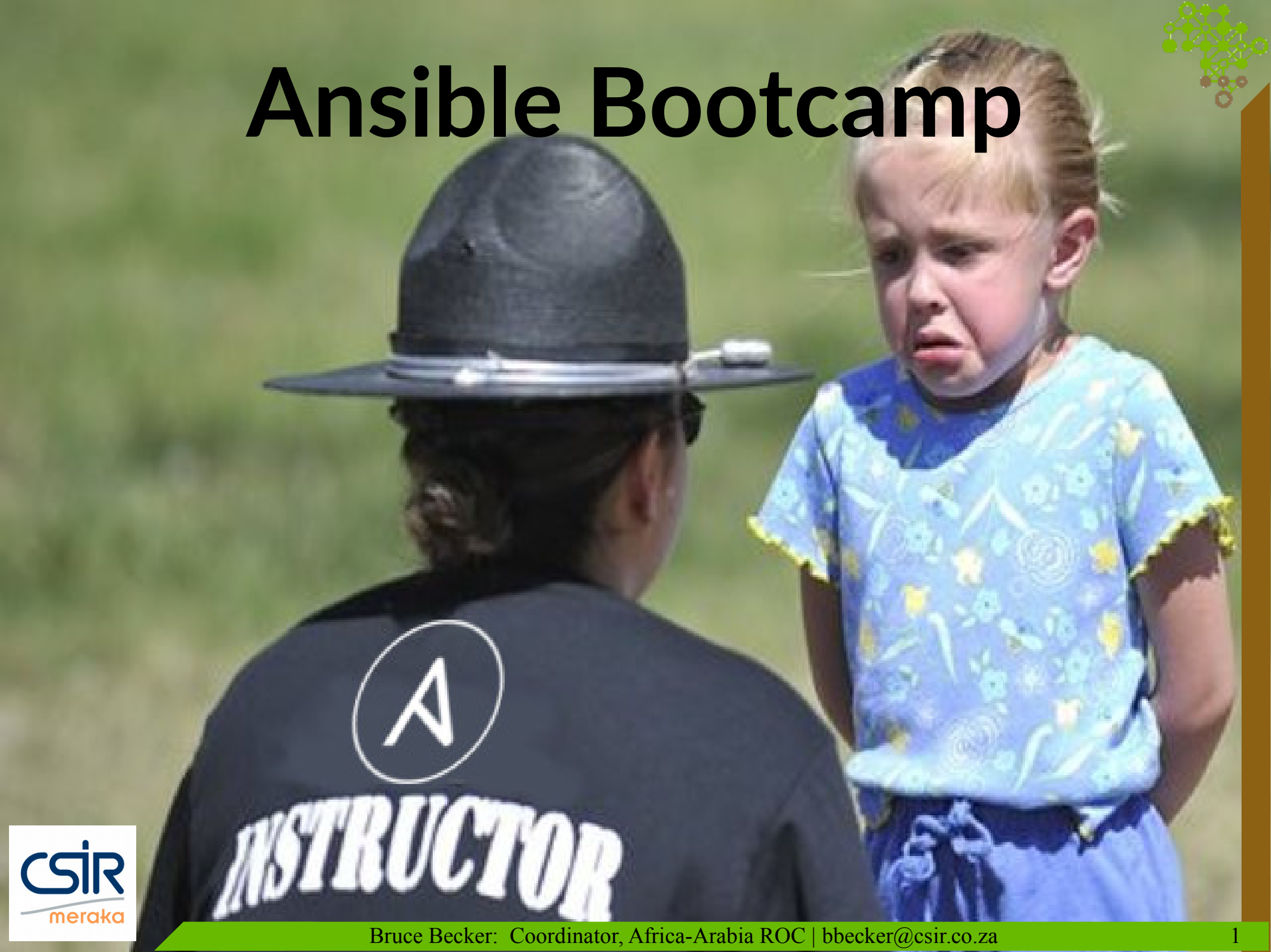
Ansible Bootcamp



Learning Goals

- Explain what Ansible is (What)
- Describe Ansible use cases (Why)
- Identify use cases and describe the solutions Ansible provide (When)
- Know the components of Ansible (How)

Ansible

- Ansible is for :
  - Application Deployment
  - Multi-Tier Orchestration
  - Configuration Management
- Why Ansible ? Ansible is :
  - Simple
  - Easy to write, read, maintain and evolve- without writing scripts or custom code
  - Fast to learn and set up

Ansible : Use Cases

- Remote execution
- Configuration Management
- Deployment and Orchestration

Ansible: Remote Execution

- Replacement for traditional systems administration tasks
- Checking system responsiveness and uptime
- Gathering information about a collection of systems
- Replace one off rsync scripts, fabric, or terminal multiplexing

Ansible: Configuration Management

- Lengthy, fine-grained system configuration,
  - *e.g.* adding users, ssh keys, installing and configuring services, and bootstrapping systems ***to a given state.***
- "Configuration remediation" – ensure consistent server configuration.
  - removes manual configuration errors
  - Checks for ”Configuration Drift”
- Version-controlled – service configurations can be reliably reproduced.

Ansible: Deployment and Orchestration

- Separate physical infrastructure from tasks
- Allow model-based description of services
- Good for complex, interdependent service deployment

Aspects of Ansible

- Simple connection model :
  - No master-slave relationships
  - Everything can configure everything (even self-configuration)
  - No custom agent to set up
  - Security provided by SSH
  - Simplest mode is *push* from control node to controlled nodes

Vocabulary

- Inventory:
  - Hosts
  - Groups
- Execution:
  - Tasks
  - Plays
  - Playbooks
- Batteries:
  - Modules
  - Library
- Funky tools
  - Ansible Galaxy
  - Ansible Container
  - Ansible Tower

Vocabulary

- Inventory:
  - Hosts
  - Groups
- Execution:
  - Tasks
  - Plays
  - Playbooks
- Batteries:
  - Modules
  - Library
- ~~Funky tools~~
  - ~~Ansible Galaxy~~
  - ~~Ansible Container~~
  - ~~Ansible Tower~~

Vocabulary: Inventory

- Hosts
  - A remote machine that Ansible manages
  - Can have individual variables (host name, connection port number, etc)
- Groups
  - Easy way to combine hosts with similar aspects
  - All hosts in a group share group variables
- Inventory
  - Fill description of hosts which you own and control
  - Can nest groups and move hosts in and out of groups

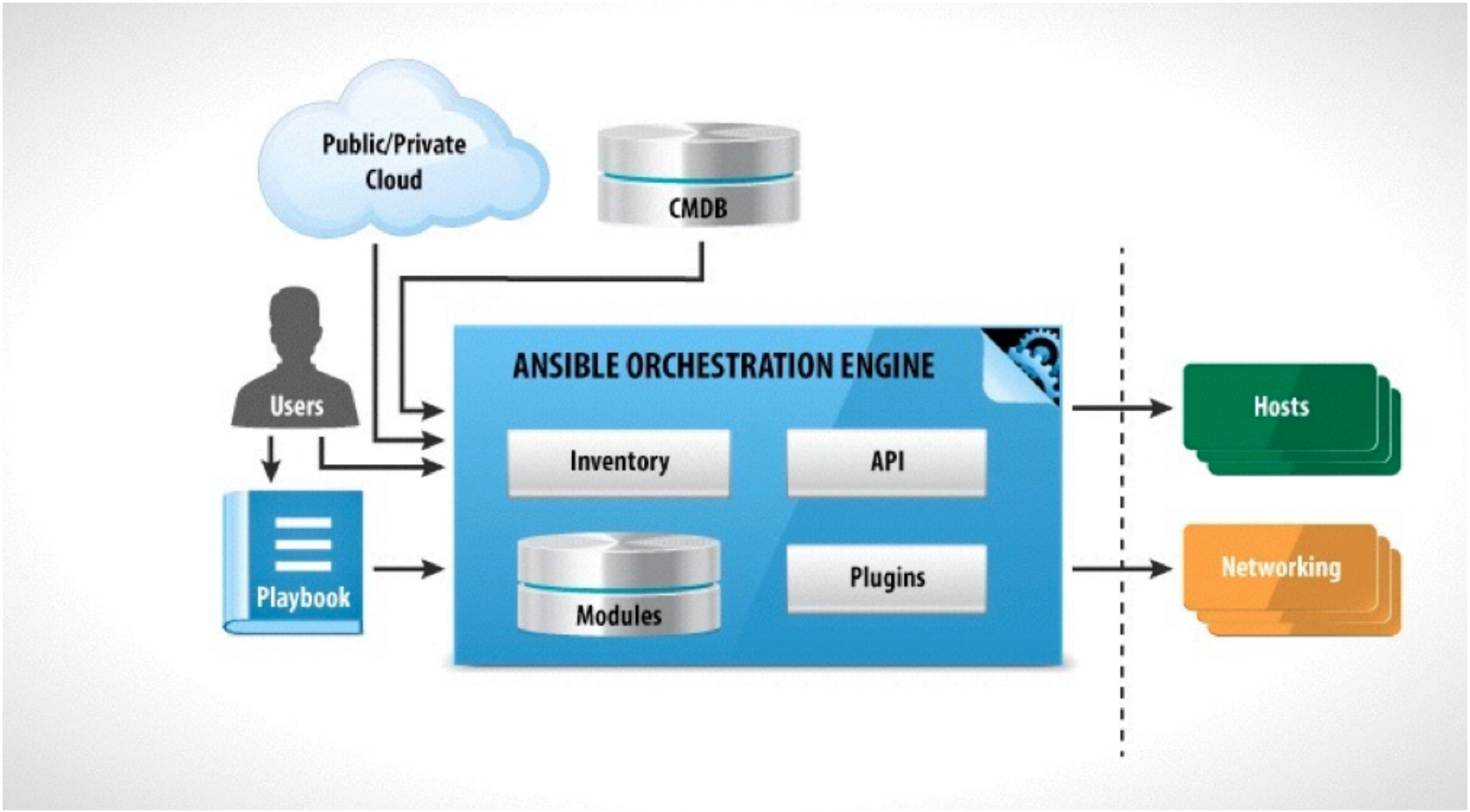
Vocabulary: Execution

- Playbook
  - How Ansible orchestrates, configures, administers or deploys systems
  - A combined, coherent description of a complex state of applications, hosts, services.
- Plays
  - A mapping between a set of hosts and the tasks which run on those hosts
  - defines the role that those systems will perform.
  - There can be one or many plays in a playbook
- Tasks: Application of a single module on the specified host(s)

Vocabulary: Batteries

- Modules:
  - units of work that Ansible ships out to remote machines.
  - can be implemented in any language.
  - Return JSON or simple `key=value` pairs.
  - Once modules are executed on remote machines, they are removed, → no long running daemons used
- Library:
  - Collection of available modules

Architecture



http://slides.com/racku/ansible-fundamentals#/4/17

Not Covered Yet

- Variables
- Filters
- Conditionals
- Loops

Ansible: Footprint

- Control machine:
  - *NIX (no Windows yet)
  - (Optional) OpenSSH Client (other ssh permitted)
  - Python 2.7
  - Since 2.0 dependencies are maintained with pip
- Managed node:
  - OpenSSH server
  - > Python-2.7

Hands-on session
Getting to know Ansible

You will need

- Your dedicated machine already knows kung-fu – it has been configured to have Ansible (by Ansible)
- We will run some ad-hoc commands with Ansible to get a feel for it. Once you've logged in:
  - What version of python do you have ?
  - Where is Ansible installed ?
  - What version of Ansible is installed ?

Ansible Ad-Hoc

Refer to Ansible documentation :
http://docs.ansible.com/ansible

1. Create an inventory of localhost
2. Collect facts
3. Ensure that there is an Ansible user on the host
4. Ensure that ntpd is installed

Ansible Ad-Hoc

See Ansible Inventories

1. Create your project : `mkdir ~/entebbe-workshop`
2. Create the repo on github where you will be working
    1. Add README and LICENSE !
3. Start the repo :

```
cd ~/entebbe-workshop
git init
git remote add origin https://github.com/<yourname>/<your repo>
git pull origin master
```

4. Create a file : `inventory.local`:

```
[local]
localhost
```

Ansible Ad-Hoc

Refer to Ansible documentation :
http://docs.ansible.com/ansible

1. ~~Create an inventory of localhost~~

```
git add inventory.local
git commit -m ”Added basic local inventory”
git push origin master
```

2. Collect facts
    1. Which module is used to find facts ?
    2. Trick – no inventory is needed when using localhost

```
ansible localhost -c local …
```

3. Check facts :
    1. What is your IPv4 Address ?
    2. What kind of virtualisation are you on ?

Ansible Ad-Hoc

Refer to Ansible documentation :
http://docs.ansible.com/ansible

1. ~~Create an inventory of localhost~~
2. ~~Collect facts~~
3. Ensure that there is an Ansible user on the host
    1. Which Ansible module deals with users ?
        1. Another trick : all Ansible documentation is provided locally
        2. `ansible-doc -l`
    2. Pass parameters to the module : `-a "key=value"`
    3. Set the home directory and password, generate ssh key

Ansible Ad-Hoc

Refer to Ansible documentation :
http://docs.ansible.com/ansible

1. ~~Create an inventory of localhost~~
2. ~~Collect facts~~
3. ~~Ensure that there is an Ansible user on the host~~
4. Ensure that ntpd is installed
    1. Which module ?
    2. Extra credit :
        1. What configuration is necessary to ntp ?
        2. How would you do that the ”DevOps way” ?

Up Next : Deeper into Ansible

- We have seen how Ansible can be used as a ”drop-in” replacement for shell scripts – but …
- We are still doing things *ephemerally*. We need to ensure
  - Change-control
  - Reproducibility
  - Collaboratoon with peers
- Old way : Write a long script
- DevOps way : Put configuration into a repository