CNCF Cross-Cloud CI

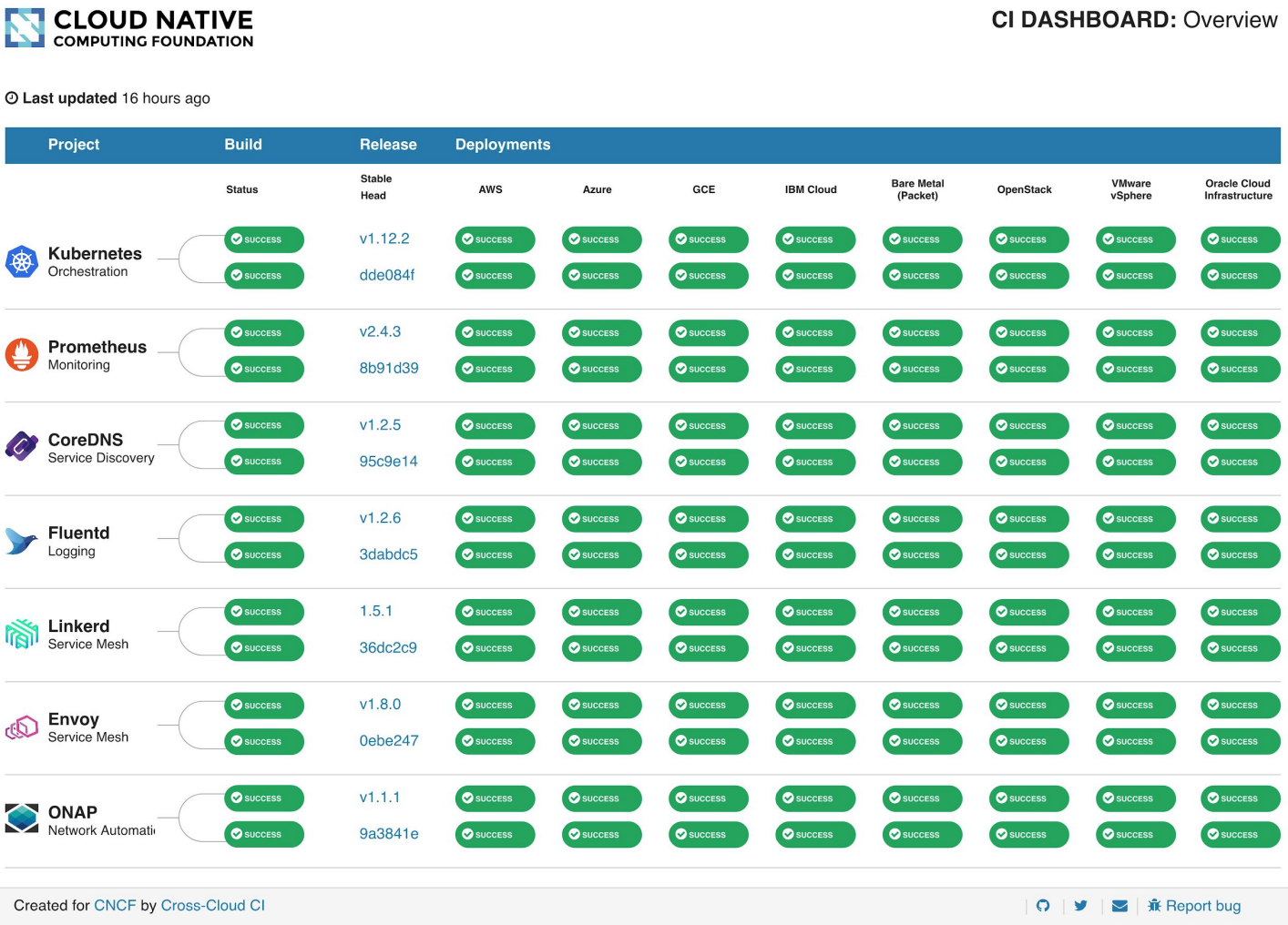
Adding support for new platforms

Agenda

- What is CNCF Cross-cloud CI?
- The Cross-cloud project
- Adding support for new platforms
- Provisioning a Kubernetes cluster
- Resolving common issues
- Ask the audience
- Contact information

What is CNCF Cross-cloud CI?

CNCF Cross-cloud CI

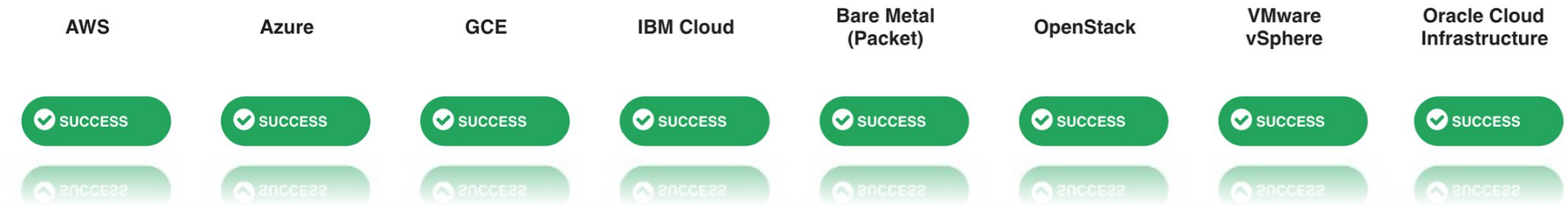


- Builds & provisions Kubernetes along with several CNCF projects to multiple platforms
- Results available on the dashboard at https://cncf.ci

The cross-cloud project

The cross-cloud project

- Cross-cloud CI is really three different components: build, cross-cloud, and cross-project
- This presentation focuses on the *cross-cloud* component
- The cross-cloud project is what enables multi-platform support



The cross-cloud project (ctd.)

- The cross-cloud project leverages Terraform inside of a container image named `provisioning`
- Directories at the root of the cross-cloud project map to platforms such as AWS, GCE, vSphere, etc.
- The platform directories are Terraform projects
- Other directories are Terraform modules, used by the platform projects to provision K8s and its dependencies

Adding support for new platforms

Adding new platforms

- Adding support for a new platform is as easy as 1..2..3..n-1
- Assuming certain requirements are met:
  - Is there a Terraform provider for the platform?
  - Is Docker installed locally?
  - An IDE with support for Terraform syntax highlighting is a plus
- Experience with Terraform is useful, but not required. Without experience, there may be a slight learning curve

Adding new platforms (ctd.)

1. Fork the cross-cloud project on GitHub
2. Clone the fork:

```
$ git clone https://github.com/akutz/cross-cloud
```

3. Add the upstream repository as a remote:

```
$ git remote add upstream https://github.com/crosscloudci/cross-cloud
```

Adding new platforms (ctd.)

4. Create the platform directory from the skeleton:

```
$ curl -sSL http://bit.ly/new-cross-cloud-platform-provider | sh -s -- KubeCon
```

The skeleton includes initial documentation, barebone Terraform files, and finally, some helper scripts for deploying and destroying clusters in the `hack` directory

Adding new platforms (ctd.)

5. Configure the platform's Terraform provider in the file `providers.tf`:

```
provider "kubecon" {
  host = "${var.host}"
  user = "${var.user}"
  pass = "${var.pass}"
}
```

6. The `${var.}` placeholders in the file above are Terraform variables and are defined in the file `input.tf`

Adding new platforms (ctd.)

7. The file `modules.tf` is responsible for loading both platform-specific modules and common modules found at the root of the cross-cloud project
    a. The platform-specific modules are responsible for creating the machine infrastructure to which the K8s cluster is deployed
    b. The common modules are used to generate x509 certificates, deploy K8s dependencies such as etcd, and ultimately deploy K8s itself

Adding new platforms (ctd.)

8. Configure the platform's K8s cloud provider:
    a. If no cloud provider is used, then this step may be ignored
    b. The cloud provider consists of two files:
        i. The cloud provider configuration template, `cloud.conf`
        ii. The Terraform file that interpolates the template, `cloud.tf`

Adding new platforms (ctd.)

9. Update the file `provision.sh` located at the root of the project with a new section for the new platform

```
# Begin kubecon
elif [[ "$CLOUD_CMD" = "kubecon-deploy" || \
        "$CLOUD_CMD" = "kubecon-destroy" ]] ; then
  ...

# End kubecon
```

10. Update the `Dockerfile` located at the root of the project so that it includes the new platform directory

```
COPY kubecon/ /cncf/kubecon/
```

Provisioning a Kubernetes cluster

Provisioning Kubernetes

- Build the cross-cloud image locally with Docker:

```
$ docker build -t provisioning .
```

- Deploy a new Kubernetes cluster:

```
$ docker run --rm -it --dns 147.75.69.23 --dns 8.8.8.8 \
  -v $(pwd)/data:/cncf/data \
  -e BACKEND=file \
  -e CLOUD=vsphere \
  -e COMMAND=deploy \
  -e NAME=kubecon \
  --env-file="${ENV_FILE}" \
  provisioning
```

- A demo of cross-cloud for VMware Cloud (VMC) on AWS

Provisioning Kubernetes (ctd.)

Deploying Kubernetes to vSphere with Cross-cloud (video)

Provisioning Kubernetes (ctd.)

Accessing Kubernetes on vSphere with Cross-cloud (video)

Provisioning Kubernetes (ctd.)

Destroying Kubernetes on vSphere with Cross-cloud (video)

Resolving issues

Resolving common issues

- If the container image, `provisioning`, is launched sans shared DNS in the resolution path, the deploy process may fail with a timeout error

```
$ docker run --rm -it --dns 147.75.69.23 --dns 8.8.8.8
```

- Remote access may also depend on shared DNS, use the `kubectl` wrapper (#170) to avoid this issue

Resolving common issues (ctd.)

- The cross-cloud image must be built from the root of the project, not from within a platform directory
- Do not forget, the `-t` flag for `docker run` is what makes it possible to use `ctrl-c` to cancel a container's entry point process. Forgetting this flag means `docker kill` is required to cancel an in-progress deployment

Resolving common issues (ctd.)

- The CNCF team must add the name of the new platform to the whitelist on the shared DNS server. Until this happens, the step that adds the entries on the shared, public DNS server will fail

Ask the audience

Ask the audience

- What is missing?
- What should be highlighted?
- What can be improved?
- Additional comments or questions?

Contact information

Contact information

- Contributors
  - Andrew Kutz <akutz@vmware.com>
  - Hui Luo <luoh@vmware.com>
- CNCF Cross-cloud CI
  - Repository - https://github.com/crosscloudci/cross-cloud
  - VMware provider pull requests:
    - #150, #151, #153, #154, #163, #169


KubeCon
CloudNativeCon
North America 2018
Thank you!