# Hands-on Infrastructure Automation with Terraform on AWS

Alexander Savchuk

# Creating reusable components with modules

Section 5

# Using external modules

# Local module

```
module "ecs_cluster" {
  source = "../module.ecs_cluster"
}
```

# Module sources

- Local paths
- Terraform Registry
- GitHub
- Bitbucket
- Generic Git, Mercurial repositories
- HTTP URLs
- S3 buckets

# Public Terraform Registry

**<NAMESPACE>/<NAME>/<PROVIDER>**

```
module "consul" {
  source = "hashicorp/consul/aws"
  version = "0.1.0"
}
```

# Private Terraform Registry

```
<HOSTNAME>/<NAMESPACE>/<NAME>/<PROVIDER>


module "consul" {
  source = "app.terraform.io/example-corp/k8s-cluster/azurerm"
  version = "0.1.1"
}
```

# Private Terraform Registry

```
<HOSTNAME>/<NAMESPACE>/<NAME>/<PROVIDER>


module "consul" {
  source = "app.terraform.io/example-corp/k8s-cluster/azurerm"
  version = "0.1.1"
}
```

# Version constraint expressions

- **>= 1.2.0**: version 1.2.0 or newer
- **<= 1.2.0**: version 1.2.0 or older
- **~> 1.2.0**: any non-beta version >= 1.2.0 and < 1.3.0, e.g. 1.2.X
- **~> 1.2**: any non-beta version >= 1.2.0 and < 2.0.0, e.g. 1.X.Y
- **>= 1.0.0, <= 2.0.0**: any version between 1.0.0 and 2.0.0 inclusive

# GitHub source

```
module "consul" {
  source = "github.com/hashicorp/example"
}
module "consul" {
  source = "git@github.com:hashicorp/example.git"
}
```

# GitHub source

```
module "consul" {
  source = "github.com/hashicorp/example"
}
module "consul" {
  source = "git@github.com:hashicorp/example.git"
}
```

# GitHub source

```
module "consul" {
  source = "github.com/hashicorp/example"
}
module "consul" {
  source = "git@github.com:hashicorp/example.git"
}
```

# Bitbucket source

```
module "consul" {
  source = "bitbucket.org/hashicorp/terraform-consul-aws"
}
```

# Generic Git Repository

```
module "vpc" {
  source = "git::https://example.com/vpc.git"
}
module "vpc" {
  source = "git::ssh://username@example.com/vpc.git"
}
```

# Generic Mercurial Repository

```
module "vpc" {
  source = "hg::http://example.com/vpc.hg"
}
```

# Selecting a revision

```
module "vpc" {
  source = "git::https://example.com/vpc.git?ref=v1.2.0"
}
module "vpc" {
  source = "hg::http://example.com/vpc.hg?ref=v1.2.0"
}
```

# No interpolations in the module configuration

# Modules in sub-directories

```
module "vpc" {
  source =
"git::https://example.com/network.git//modules/vpc?ref=v1.2.0"
}
```

# Module tips

- Keep explicit provider configurations only in the root module
- Use module-relative path for embedded files (`${path.module}`)
- Expose ids of resources as outputs, unless you want to keep them hidden
- Avoid using inline blocks where possible
- Avoid nesting modules
- Find a balance between flexibility and readability
- Leverage modules from Terraform registry to avoid reinventing the wheel
- Specify exact module version to prevent unexpected updates

# Module tips

- Keep explicit provider configurations only in the root module
- Use module-relative path for embedded files (`${path.module}`)
- Expose ids of resources as outputs, unless you want to keep them hidden
- Avoid using inline blocks where possible
- **Avoid nesting modules**
- Find a balance between flexibility and readability
- Leverage modules from Terraform registry to avoid reinventing the wheel
- Specify exact module version to prevent unexpected updates

# Managing multiple environments

Next Section