



State Management and Provisioners

Subtitle or speaker name



Recap

Bigger picture around IaC

Terraform

- Core Workflow
- Config Options, Expressions, Functions
- Dependency Management and Imports
- Local and Remote Modules
- State Management
 - Remote State Management using Azure Blob Storage
- Niche Topics
 - Workspaces and Provisioners
- DevOps with Terraform

Conditions and Terms of Use

Microsoft Confidential

This training package is proprietary and confidential, and is intended only for uses described in the training materials. Content and software is provided to you under a Non-Disclosure Agreement and cannot be distributed. Copying or disclosing all or any portion of the content and/or software included in such packages is strictly prohibited.

The contents of this package are for informational and training purposes only and are provided "as is" without warranty of any kind, whether express or implied, including but not limited to the implied warranties of merchantability, fitness for a particular purpose, and non-infringement.

Training package content, including URLs and other Internet website references, is subject to change without notice. Because Microsoft must respond to changing market conditions, the content should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication. Unless otherwise noted, the companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred.

Copyright and Trademarks

© 2016 Microsoft Corporation. All rights reserved.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

For more information, see **Use of Microsoft Copyrighted Content** at
<https://www.microsoft.com/en-us/legal/intellectualproperty/permissions/default.aspx>

Microsoft®, Internet Explorer®, Outlook®, SkyDrive®, Windows Vista®, Zune®, Xbox 360®, DirectX®, Windows Server® and Windows® are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Other Microsoft products mentioned herein may be either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. All other trademarks are property of their respective owners.

Remote State

Terraform State

Terraform must store state about your managed infrastructure and configuration.



Local
State

Remote
State

Local state is fragile and not fit for production. (or anything outside local environment)

State File Fundamentals

- State files are required to manage a resource past day 1
- State files can contain secrets
- Local state files cannot be used for a team size of >1
- Continuous Delivery pipelines need access to state files
- State file locking is important
- Granular permissions are not supported with open source Terraform
 - If outputs are required, full state file access is required
- State files contain multi-cloud data that can be mined

State File Scaling and Security

- Remote State must be used
- For Azure: Blob Storage or Terraform Cloud / Enterprise
- Blob Storage structure
 - Single storage account with a single container
 - Single storage account with multiple containers
 - Storage account per subscription, resource group or resource
- Terraform Cloud / Enterprise granular security
 - View runs, but no access to state file
 - Only access to outputs and not full state file
- Use principle of least privilege and automate creation
- Use Zero-Trust: OpenID Connect or Managed Service Identity

terraform backend block - remote state setup for community version

```
# main.tf
terraform {
  backend "azurerm" {
    resource_group_name    = "<rg-name>"
    storage_account_name   = "<sg-acct-name>"
    container_name         = "<container-name>"
    key                    = "terraform.tfstate" # default state file name
  }
}
```

Running an init will create a remote backend.

```
terraform init
```


Partial config with terraform backend

```
# main.tf
terraform {
  backend "azurerm" {
    resource_group_name = "<rg-name>"
    container_name       = "<container-name>"
    key                  = "terraform.tfstate" # default state file name
  }
}
```

```
# Pass the partial config args during the init
terraform init -backend-config="storage_account_name=<sg-account-name>"
```

Remote State using Azure Storage

Lab 7

Local Workspaces

Terraform Workspaces

Provides a way to separate out **.tfstate** files by their own space/directory

Similar to feature branches in version-control


Provides isolation during plan, apply and destroy

Initially, terraform creates just one workspace called "default"

Remote backends such as AzureRM support having multiple workspaces


Workspaces does not warrant complete isolation use-cases. See <https://www.terraform.io/docs/state/workspaces.html#when-to-use-multiple-workspaces>

Workspace Management




```
▶ terraform workspace new my-experimental-workspace  
Created and switched to workspace "my-experimental-workspace"!
```


You're now on a new, empty workspace. Workspaces isolate their state, so if you run "terraform plan" Terraform will not see any existing state for this configuration.



```
▶ terraform workspace list  
default  
* my-experimental-workspace  
my-workspace
```



```
▶ terraform workspace select my-workspace  
Switched to workspace "my-workspace".
```



```
▶ tree terraform.tfstate.d  
terraform.tfstate.d  
├── my-experimental-workspace  
│   └── terraform.tfstate  
└── my-workspace  
    └── terraform.tfstate  
  
2 directories, 2 files
```

Provisioners

Provisioners are a Last Resort

Can be used to model specific actions on the local machine or on a remote machine in order to prepare servers or other infrastructure objects for service. (can also be used with *terraform_data*)

Introduced in terraform as a measure of pragmatism, knowing that there will always be certain behaviours that can't be directly represented in Terraform's declarative model.

Built-in Provisioners



Local-
exec

Remote-
exec

Use Provisioners as a last resort as they add complexity and uncertainty.

Ask

Discuss

Comment

