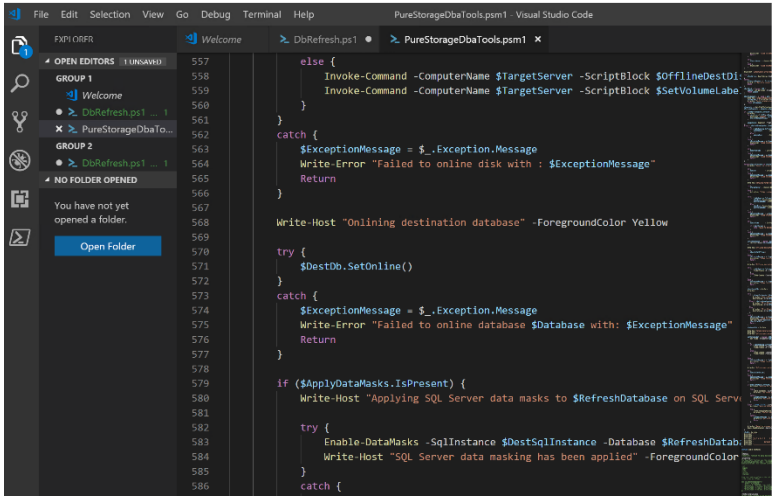
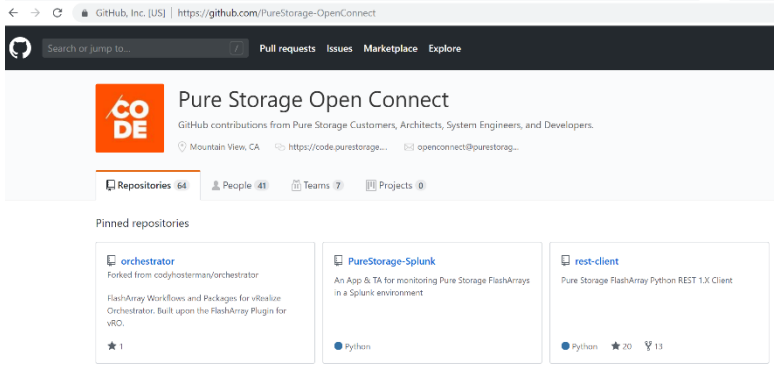
SQL SERVER DATABASE DEVOPS

PURESTORAGE®

HOW DEVELOPERS VIEW THE WORLD

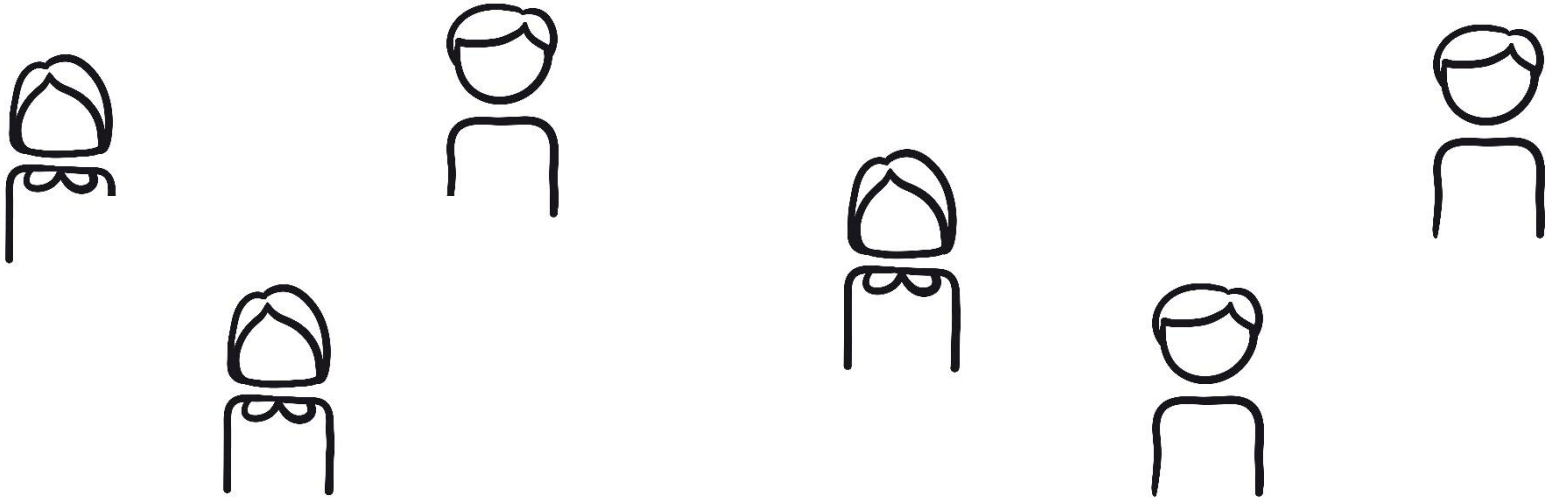




developers also tend to work in teams . . .

PURESTORAGE®

TEAM BASED SOFTWARE DEVELOPMENT



How do you **integrate** changes from multiple developers into a single coherent piece of software that works ?

CONTINUOUS INTEGRATION (CI)

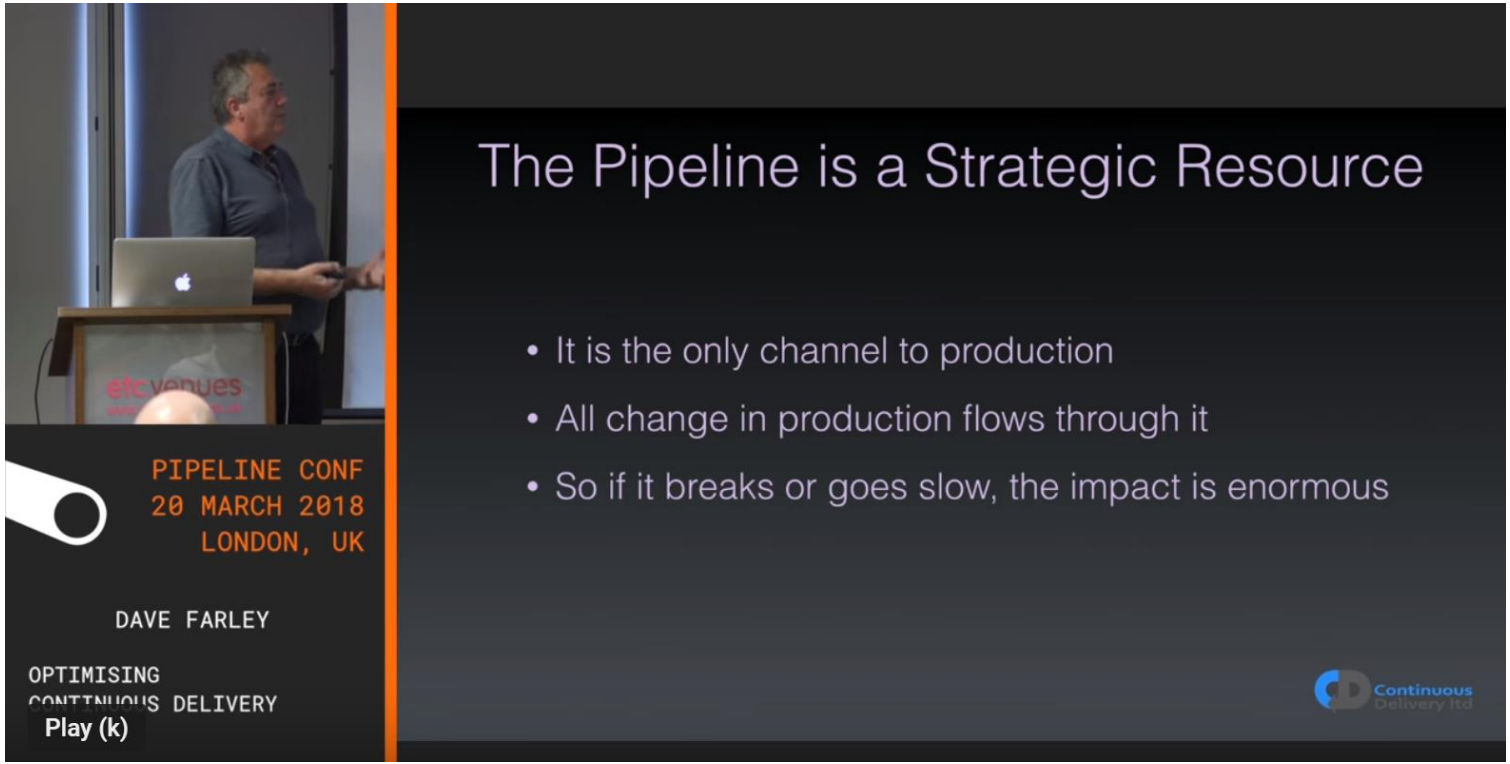
Continuous Integration (CI) is a development practice that requires developers to integrate code into a shared repository several times a day. Each check-in is then verified by an automated build, allowing teams to detect problems early.

CI PIPELINE 101

1. Check out code
2. Build artifact
3. Deploy artifact to target
4. Run tests

PURESTORAGE®

WHY DO CI PIPELINES MATTER ?



PURESTORAGE®

SOMETHING NEEDS TO RUN THE PIPELINE

Continuous integration engines

circleci

PURESTORAGE®

PIPELINE AS CODE, BUT WHY ?

Which approach leads to better reusability, extensibility, repeatable results and better automation ?

“TALK IS CHEAP, SHOW ME THE CODE”

LINUS TORVALDS
CREATOR OF LINUX AND GIT

STORING THE PIPELINE WITH THE SOURCE CODE

PureStorage-OpenConnect / Jenkins-Fa-Snapshot-Ci-Pipeline-Adv

Unwatch 8 | Star 0 | Fork 0

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Insights | Settings

Jenkins CI pipeline which illustrates testing and database refreshes in parallel

Edit

Manage topics

28 commits | 1 branch | 0 releases | 1 contributor | Apache-2.0

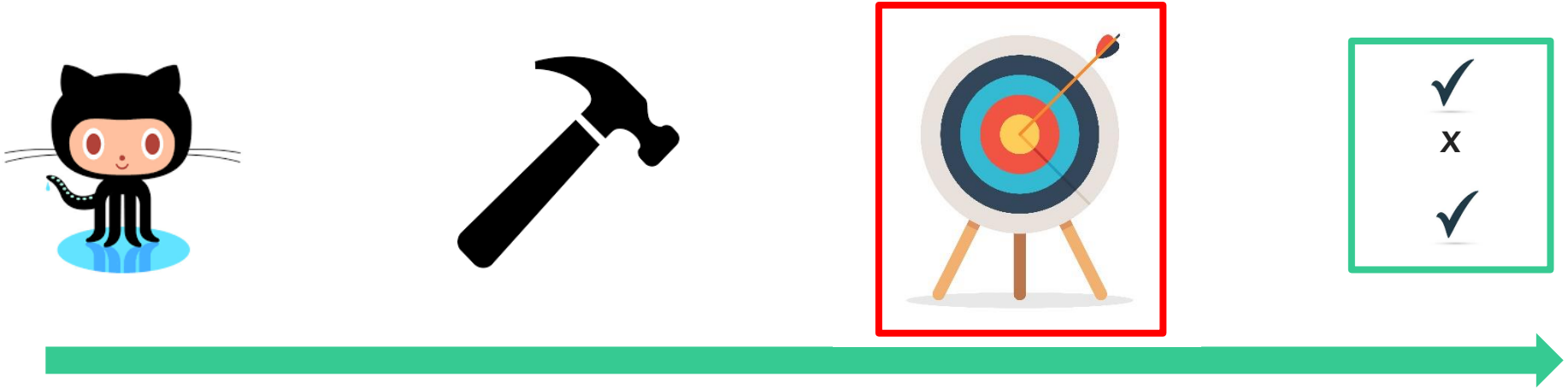
Branch: master | New pull request | Create new file | Upload files | Find File | Clone or download

chrisadkin Update Jenkinsfile — Latest commit f10c15e 18 hours ago

.vs/Jenkins-Fa-Snapshot-Ci-Pipeline-Adv/v15	Initial commit	a day ago
Jenkins-Fa-Snapshot-Ci-Pipeline-Adv	sqlproj file correction	a day ago
Jenkins-Fa-Snapshot-Ci-Pipeline-Adv.sln	Initial commit	a day ago
Jenkinsfile	Update Jenkinsfile	18 hours ago
LICENSE	Initial commit	a day ago
README.md	Initial commit	a day ago

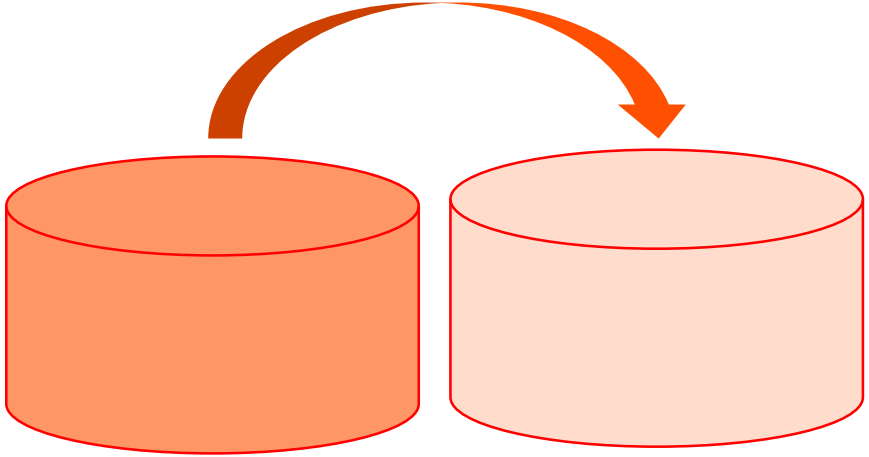
PURESTORAGE®

A COMMON PROBLEM



- Deployment target data differs from production data
- Deployment target is slow and cumbersome to provision or refresh

A SOLUTION



Use a snapshot based database refresh to refresh the target

WRITING SCRIPTS, A GOOD ANALOGY

A MODULE TO DO THE HEAVY LIFTING

```
Install-Module –Name PureStorageDbaTools

Get-Command –Module PureStorageDbaTools

Command Type   Name
------------   ----------------------------
Function       Invoke-DynamicDataMasking
Function       Invoke-PfaDbRefresh
Function       Invoke-StaticDataMasking
Function       Invoke-PfaDbSnapshot
```

FULLY SELF DOCUMENTED

```
Get-Help [Function]

Get-Help [Function] –Detailed

Get-Help [Function] –Full

Get-Help [Function] –Examples
```

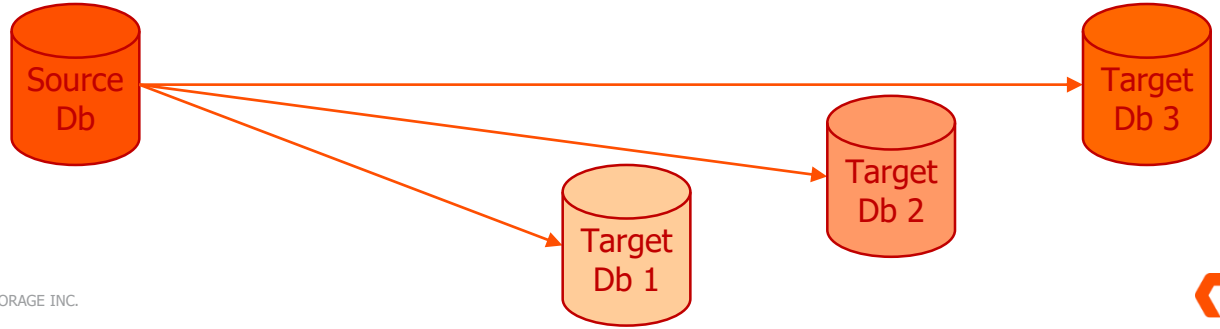
SECURITY

```
$Pwd = Get-Content ‘C:\Temp\Secure-Credentials.txt’ | ConvertTo-SecureString
$Creds = New-Object System.Management.Automation.PSCredential("pureuser", $pwd)
Invoke-PfaDbRefresh -RefreshDatabase tpch-no-compression `
                    -RefreshSource   Z-STN-WIN2016-A\DEVOPSPRD `
                    -DestSqlInstance Z-STN-WIN2016-A\DEVOPSIAT `
                    -PfaEndpoint     10.225.112.10 `
                    -PfaCredentials  $Creds
```

PURESTORAGE®

1:M DATABASE REFRESHES

```
$Pwd = Get-Content ‘C:\Temp\Secure-Credentials.txt’ | ConvertTo-SecureString
$Creds = New-Object System.Management.Automation.PSCredential("pureuser", `
                                                                  $pwd)
$Targets = @('Z-STN-WIN2016-A\DEVOPSDEV1’, `
             'Z-STN-WIN2016-A\DEVOPSDEV2’, `
             'Z-STN-WIN2016-A\DEVOPSDEV3’)
Invoke-PfaDbRefresh -RefreshDatabase tpch-no-compression `
                    -RefreshSource   Z-STN-WIN2016-A\DEVOPSPRD `
                    -DestSqlInstance $Targets `
                    -PfaEndpoint     10.225.112.10 `
                    -PfaCredentials  $Creds
```



PURESTORAGE®

SNAPSHOT A DATABASE

```
$Pwd = Get-Content ‘C:\Temp\Secure-Credentials.txt’ | ConvertTo-SecureString
$Creds = New-Object System.Management.Automation.PSCredential("pureuser", `
                                                                $pwd)
New-PfaDbRefresh -Database       tpch-no-compression `
                 -SqlInstance    Z-STN-WIN2016-A\DEVOPSPRD `
                 -PfaEndpoint    10.225.112.10 `
                 -PfaCredentials $Creds
```

DATA MASKING: STATIC

```
New-DbaDbMaskingConfig -SqlInstance z-sql-prd `
                       -Database tpch-no-compression `
                       -Path D:\apps\datamasks\z-sql-prd.tpch-no-compression.tables.json

StaticDataMaskFile = "D:\apps\datamasks\z-sql-prd.tpch-no-compression.tables.json"
$Targets = @("z-sql2016-devops-tst", "z-sql2016-devops-dev")
Invoke-PfaDbRefresh -$RefreshDatabase    tpch-no-compression `
                    -RefreshSource       z-sql-prd `
                    -DestSqlInstance     $Targets `
                    -PfaEndpoint         10.225.112.10 `
                    -PfaCredentials      $Creds `
                    -StaticDataMasksFile $StaticDataMaskFile
```


PURESTORAGE®

DATA MASKING: DYNAMIC

```
$Targets = @("z-sql2016-devops-tst", "z-sql2016-devops-dev")
Invoke-PfaDbRefresh -$RefreshDatabase tpch-no-compression `
                    -RefreshSource    z-sql-prd `
                    -DestSqlInstance  $Targets `
                    -PfaEndpoint      10.225.112.10 `
                    -PfaCredentials   $Creds `
                    -ApplyDataMasks
```

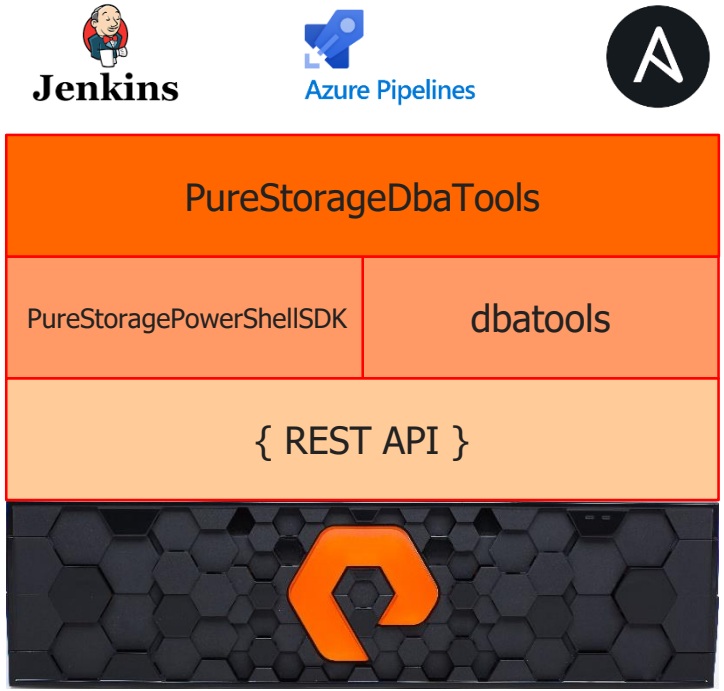
Leverages SQL Server dynamic data masking

DB REFRESH ORCHESTRATION VIA ANSIBLE

BLOG POST LINK

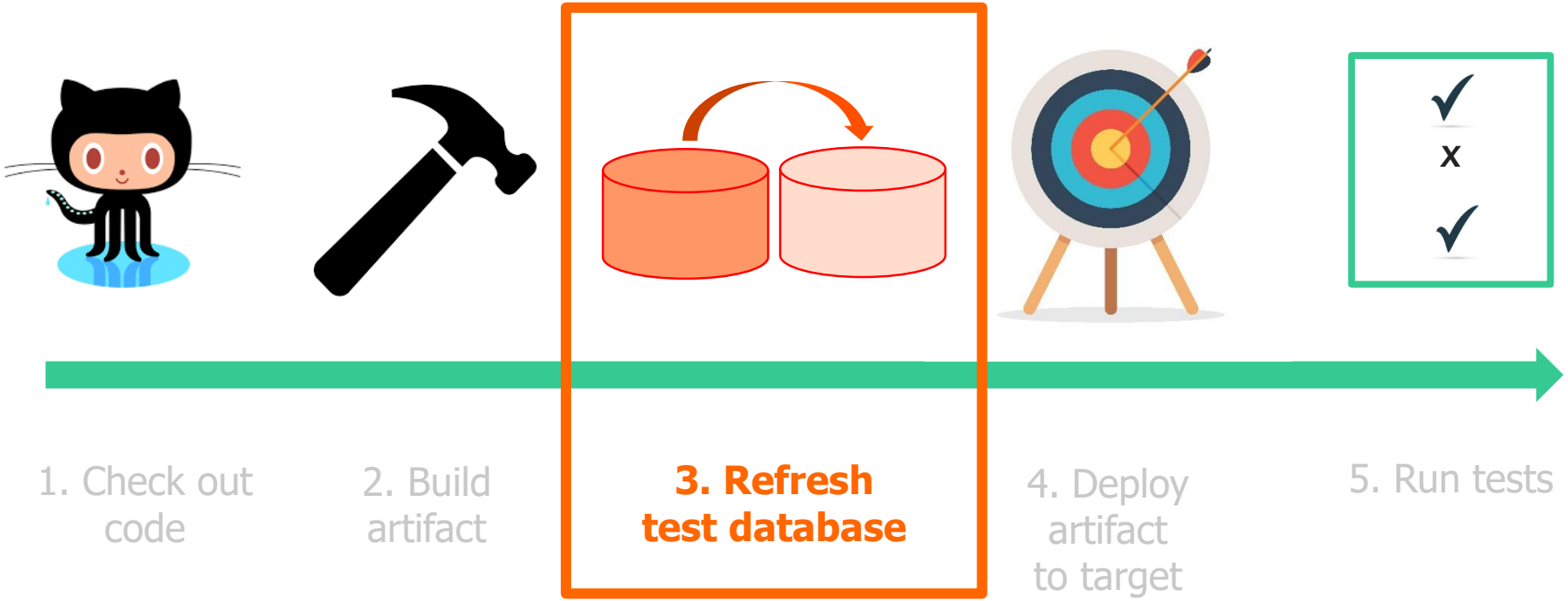
```yaml
- name: PowerShell MS SQL Database refresh
hosts: z-stn-win2016-a
gather_facts: no
vars_files:
- vars/ps.yaml

tasks:

# Include Pure Storage PowerShell DBA tools
- include: tasks/setupPS.yaml

# PowerShell database refresh
- name: PowerShell Database Snapshot
win_shell: |
$Pwd   = Get-Content 'C:\Temp\Secure-Credentials.txt' | ConvertTo-SecureString
$Creds = New-Object System.Management.Automation.PSCredential ("pureuser", $pwd)
New-PfaDbSnapshot -Database tpch-no-compression `
                  -SqlInstance {{ dbSource }}   `
                  -PfaEndpoint {{ pEndpoint }}  `
                  -PfaCredentials $Creds
args:
chg_dir: "{{ tgt_dir }}"
register: snapshot

# - debug:
# msg: "Refreshing database using snapshot name {{ snapshot }}"

- set_fact:
sName: "{{snapshot.stdout_lines | select('match','name.*') | list | first}}"

- set_fact:
snapName: "{{ sName[10:] }}"

# - debug:
# msg: "Refreshing database using snapshot name {{ snapName }}"

- name: PowerShell Database Refesh
win_shell: |
$Pwd = Get-Content 'C:\Temp\Secure-Credentials.txt' | ConvertTo-SecureString
$Creds = New-Object System.Management.Automation.PSCredential ("pureuser", $pwd)
Invoke-PfaDbRefresh -RefreshDatabase tpch-no-compression `
                    -RefreshSource   {{ snapName }}      `
                    -DestSqlInstance {{ dbTarget }}      `
                    -PfaEndpoint     {{ pEndpoint }}     `
                    -PfaCredentials  $Creds              `
                    -RefreshFromSnapshot                 `
                    -NoPsRemoting
register: refresh

- debug: msg="{{ refresh.stdout_lines }}"
```

LAYERS OF THE "SOLUTION STACK"



PURESTORAGE®

INTEGRATES EASILY INTO CI PIPELINES



PURESTORAGE®

DATABASE REFRESH WITH JENKINS

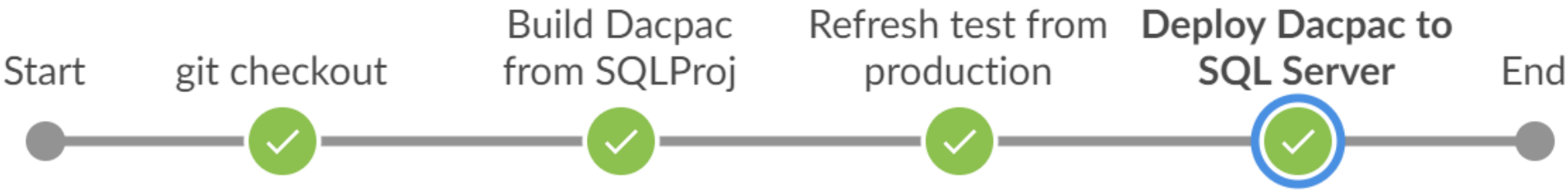
```
stage('Refresh test from production')
{
    timeout(time:5, unit:'MINUTES') {
        withCredentials([string(credentialsId: 'PfaCredentialsFile', variable: 'PfaCredentialsFile'),
                         string(credentialsId: 'PfaUser'           , variable: 'PfaUser')]) {
            powershell 'Import-Module -Name PureStorageDbaTools; ' +
                       '\$Pwd = Get-Content ' + "${PfaCredentialsFile}" + ' | ConvertTo-SecureString;'  +
                       '\$Creds = New-Object System.Management.Automation.PSCredential(\"' + "${PfaUser}" + '\",\$Pwd); ' +
                       'Invoke-PfaDbRefresh -RefreshDatabase ' + "${params.Database}"       +
                                            ' -RefreshSource   ' + "${params.SourceInstance}" +
                                            ' -DestSqlInstance ' + "${params.DestInstance}"   +
                                            ' -PfaEndpoint     ' + "${params.PfaEndpoint}"    +
                                            ' -PfaCredentials  \$Creds'
        }
    }
}
```


PURESTORAGE®

AZURE PIPELINE DATABASE REFRESH

```yaml
#
# Example Azure DevOps pipeline to:
#
# 1. Checkout a SQL Server data tools project
# 2. Build the project into a DACPAC
# 3. Refresh a development database from pseudo production, this is carried out by
#    a call to the Invoke-PfaDbRefresh function from the PureStorageDbaTools PoSH module
# 4. Apply the DACPAC to the development database
#
trigger:
- master

pool: $(agentPool)

steps:
  - task: MSBuild@1
    displayName: 'Build DACPAC'
    inputs:
      solution: 'AzureDevops-Fa-Snapshot-CI-Pipeline.sln'
      msbuildArguments: '/property:OutDir=bin\Release'
  # Create a secret variable
  - powershell: |
      $securePassword = ConvertTo-SecureString -String '$(pfaPassword)' -AsPlainText -Force
      $pfaCreds = New-Object System.Management.Automation.PSCredential '$(pfaUsername)', $securePassword
      Invoke-PfaDbRefresh -RefreshDatabase $(refreshDatabase) `
                          -RefreshSource   $(refreshSource) `
                          -DestSqlInstance $(refreshTarget) `
                          -PfaEndpoint     $(pfaEndpoint) `
                          -PfaCredentials  $pfaCreds
```

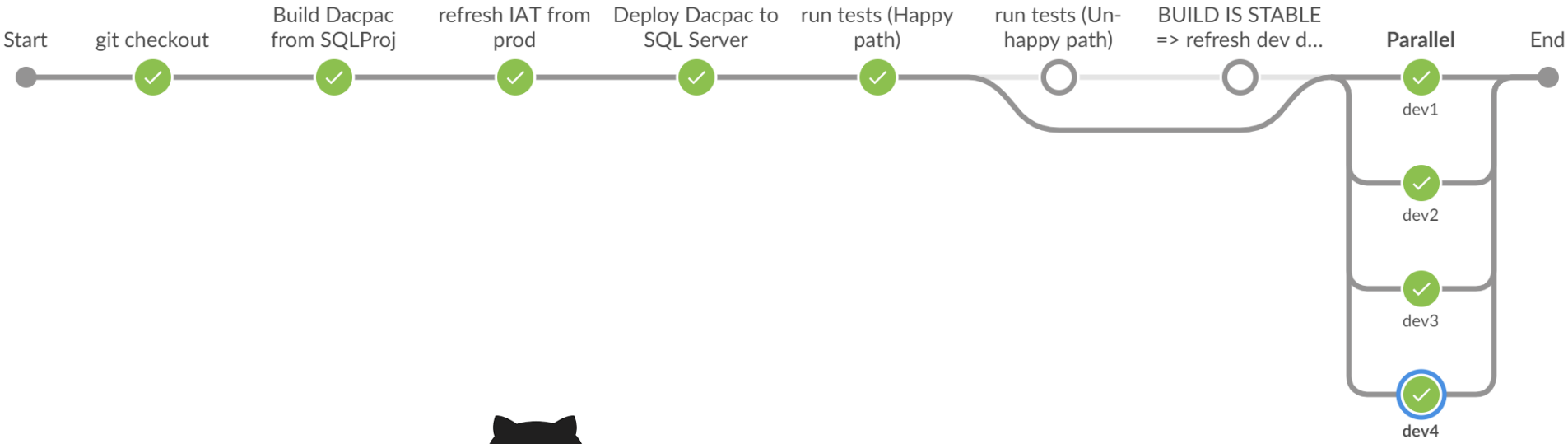
A SIMPLE EXAMPLE



GitHub repo

PURESTORAGE®

A MORE COMPLEX EXAMPLE



GitHub repo

PURESTORAGE®

Leveraging The Public Cloud

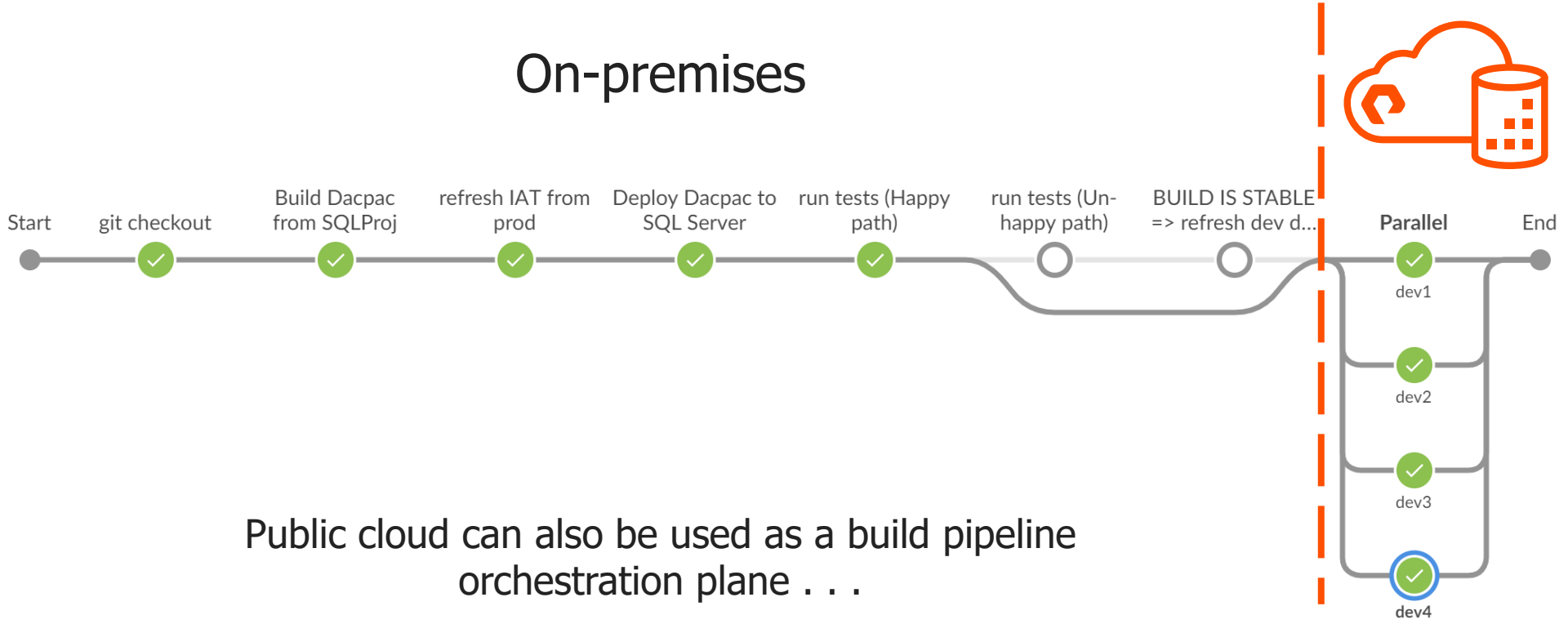
AN EXAMPLE PIPELINE

Refreshing four development database in parallel following a successful build and test

PURESTORAGE®

DEVELOPMENT DATABASES IN THE CLOUD

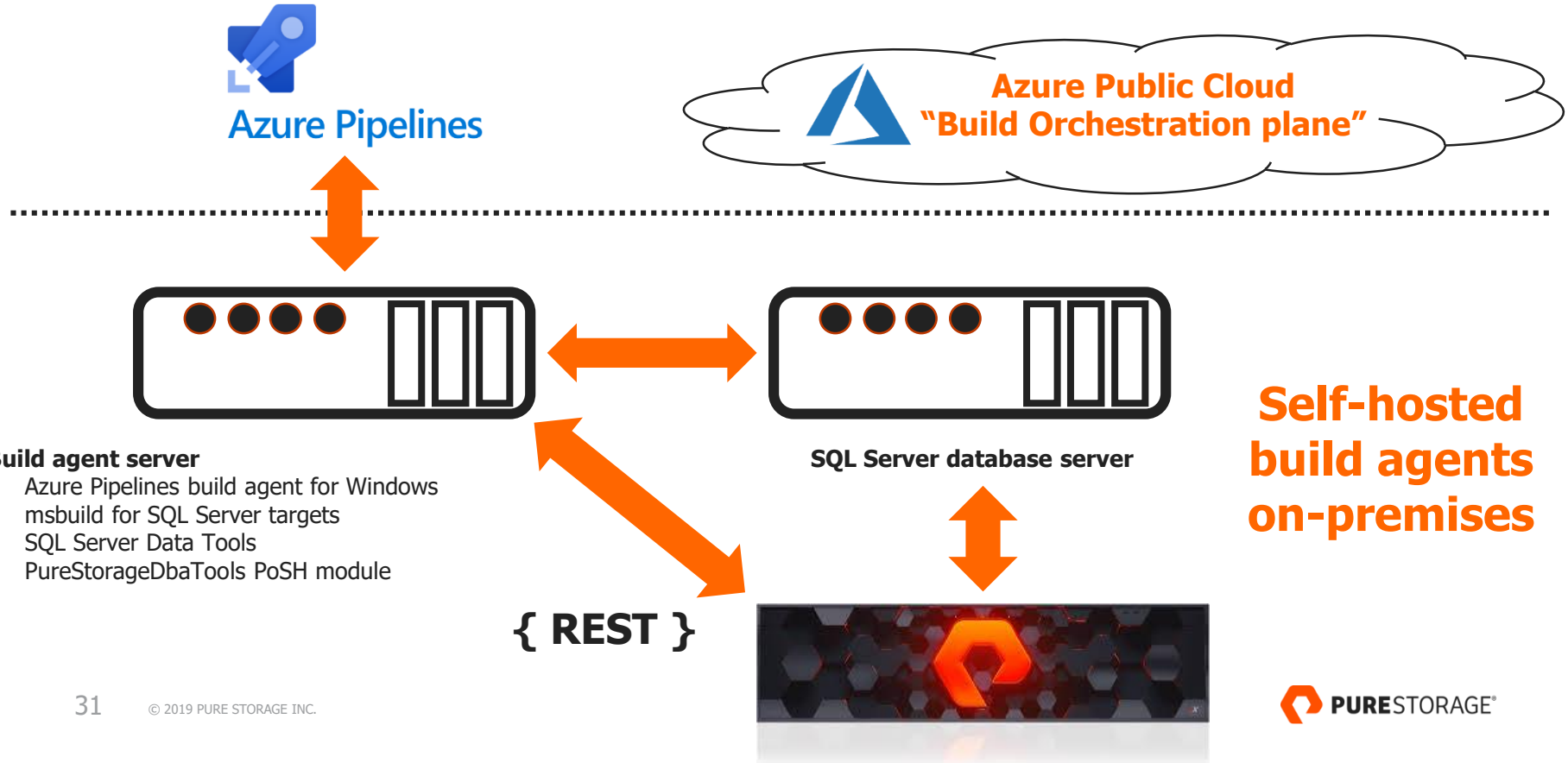
On-premises



Public cloud can also be used as a build pipeline orchestration plane . . .

PURESTORAGE®

ON-PREM PIPELINE ORCHESTRATED IN THE AZURE CLOUD

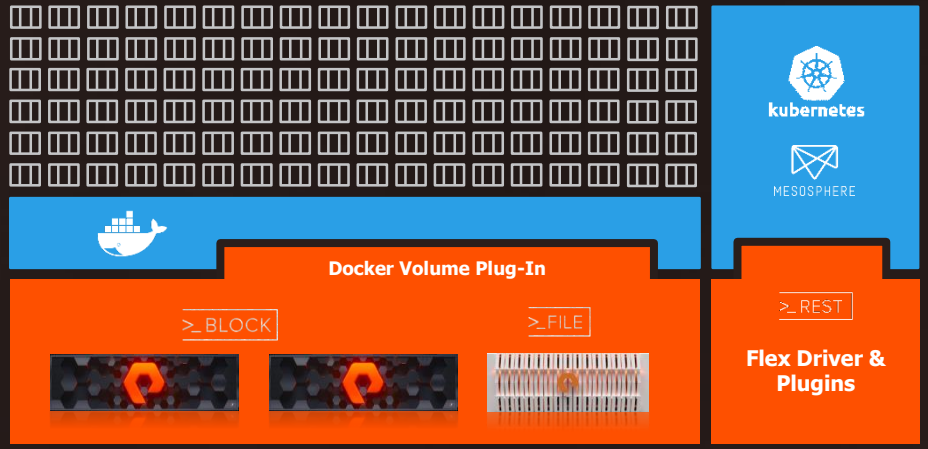


PURESTORAGE®

CONTAINERS

A GOOD ANALOGY

PURE
SERVICE ORCHESTRATOR



docker

VOLUME PLUG-IN

MESOSPHERE

ORCHESTRATION PLUG-INS

docker CERTIFIED Plugins

RED HAT OPENSHIFT PRIMED

AGENTS

- **“Build agents”**
  - Pull code from repos
  - Turn code into deployable artifacts
  - Run tests
  - Can run in a variety of ways

AGENTS AS CONTAINERS

```
pipeline {
    agent {
        docker { image 'node:7-alpine' }
    }
    stages {
        stage('Test') {
            steps {
                sh 'node --version'
            }
        }
    }
}
```

AZURE PIPELINES AGENT POOLS

https://dev.azure.com/cadkin/_settings/agentpools?poolId=9&_a=agents

Azure DevOps cadkin / Organization Settings / Agent pools Search

New Organization Permissions Settings Page: We have updated the Permissions settings user experience. Try out this Preview feature! Try it! Not now

Organization Settings

General
- Overview
- Projects
- Users
- Billing
- Global notifications
- Usage
- Extensions
- Azure Active Directory

New agent pool...

- All agent pools
  - Default
  - Hosted
  - Hosted macOS
  - Hosted macOS High Sierra
  - Hosted Ubuntu 1604
  - Hosted VS2017
  - Hosted Windows 2019 with ...
  - Hosted Windows Container
  - Self Hosted Win2016

Agents for pool Self Hosted Win2016 Download agent

Agents Roles Details Settings Maintenance history

Enabled	Name	State	Current stat
☑	Z-SQL-UTIL	Online	Idle

Requests Capabilities

ID	Type	Pipeline	Name	Date queued
162	Build	chrisadkin.AzureDevO...	20190412.2	4/12/2019 9
161	Build	PureStorage-OpenCon...	20190218.25	2/18/2019 4
160	Build	PureStorage-OpenCon...	20190218.24	2/18/2019 4
159	Build	PureStorage-OpenCon...	20190218.23	2/18/2019 4
158	Build	PureStorage-OpenCon...	20190218.22	2/18/2019 4
157	Build	PureStorage-OpenCon...	20190218.21	2/18/2019 4
156	Build	PureStorage-OpenCon...	20190218.20	2/18/2019 4

PURESTORAGE®

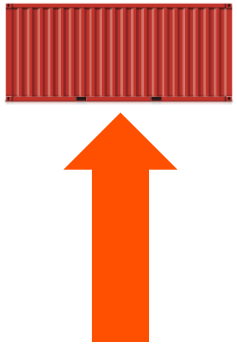
SIDECARS

"This pattern is named *Sidecar* because it resembles a sidecar attached to a motorcycle. In the pattern, the sidecar is attached to a parent application and provides supporting features for the application."

From Azure architecture patterns

‘SIDECARS’ IN A CI PIPELINE

Spin ‘Sidecar’ container deployment target up

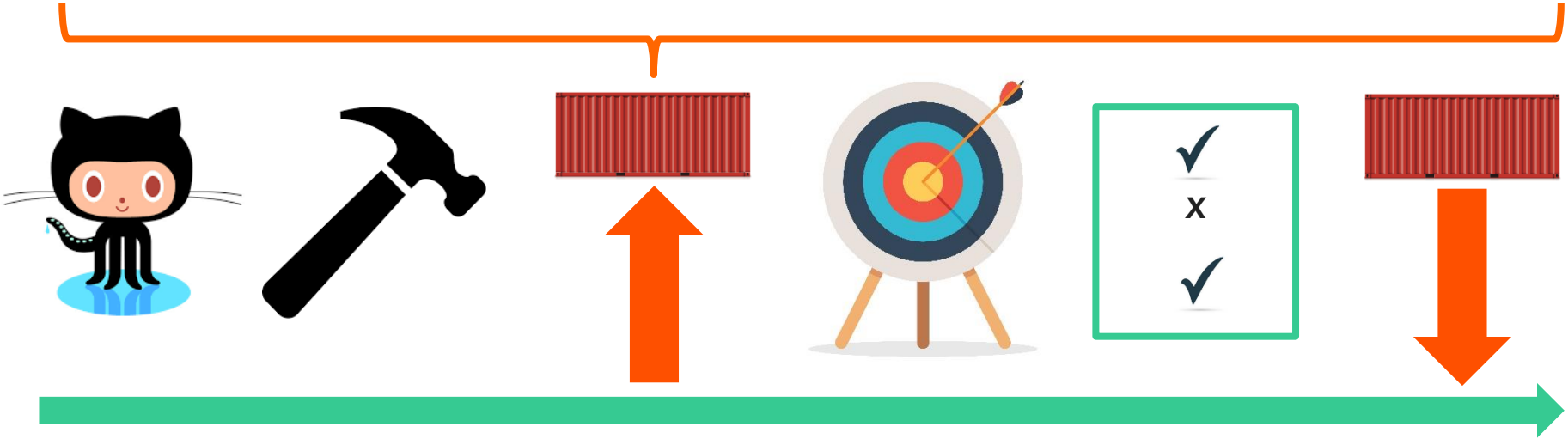


Tear ‘Sidecar’ container deployment target down

PURESTORAGE®

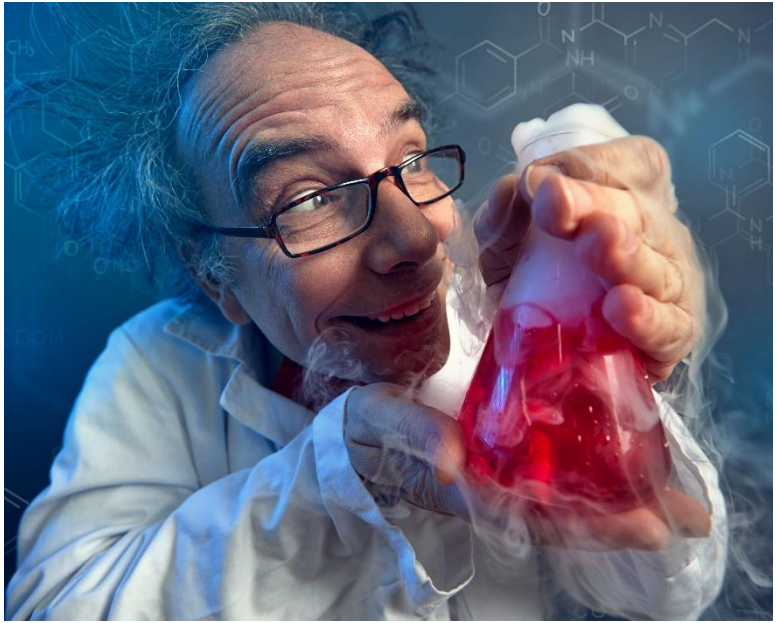
STATEFUL ‘SIDECARS’ IN A CI PIPELINE

```
docker run -v ${VOLUME_NAME}:/var/opt/mssql -e "ACCEPT_EULA=Y" -e "SA_PASSWORD=P@ssword1"
--name ${CONTAINER_NAME} -d -i -p ${PORT_NUMBER}:1433 microsoft/mssql-server-linux:2017-GA
```



PURESTORAGE®

Demonstrations



Takeaway Points

1. The CI pipeline is the conduit of software delivery from development to production,
2. How FlashArray can be leveraged in CI pipelines,
3. Value of CI-pipelines-as-code,
3. REST API as a starting point for solutions,
4. Patterns for leveraging the public cloud in CI pipelines,
5. Use of containers and PSO in CI pipelines.

USEFUL RESOURCES

- PureStorageDbaTools – available from the PowerShell gallery
- Orchestrating SQL Server Database Refreshes with Ansible
- Jenkins FlashArray Db Refresh 'Basic' Pipeline
- Jenkins FlashArray Pipeline with parallel dev database refresh
- Azure DevOps FlashArray Pipeline
- Pure Service Orchestrator