Agile infrastructure

(IN AZURE)

Speaker presentation

- Johan Kardell
- .net developer since 2004
- Devops since 2006
- Azure since 2014

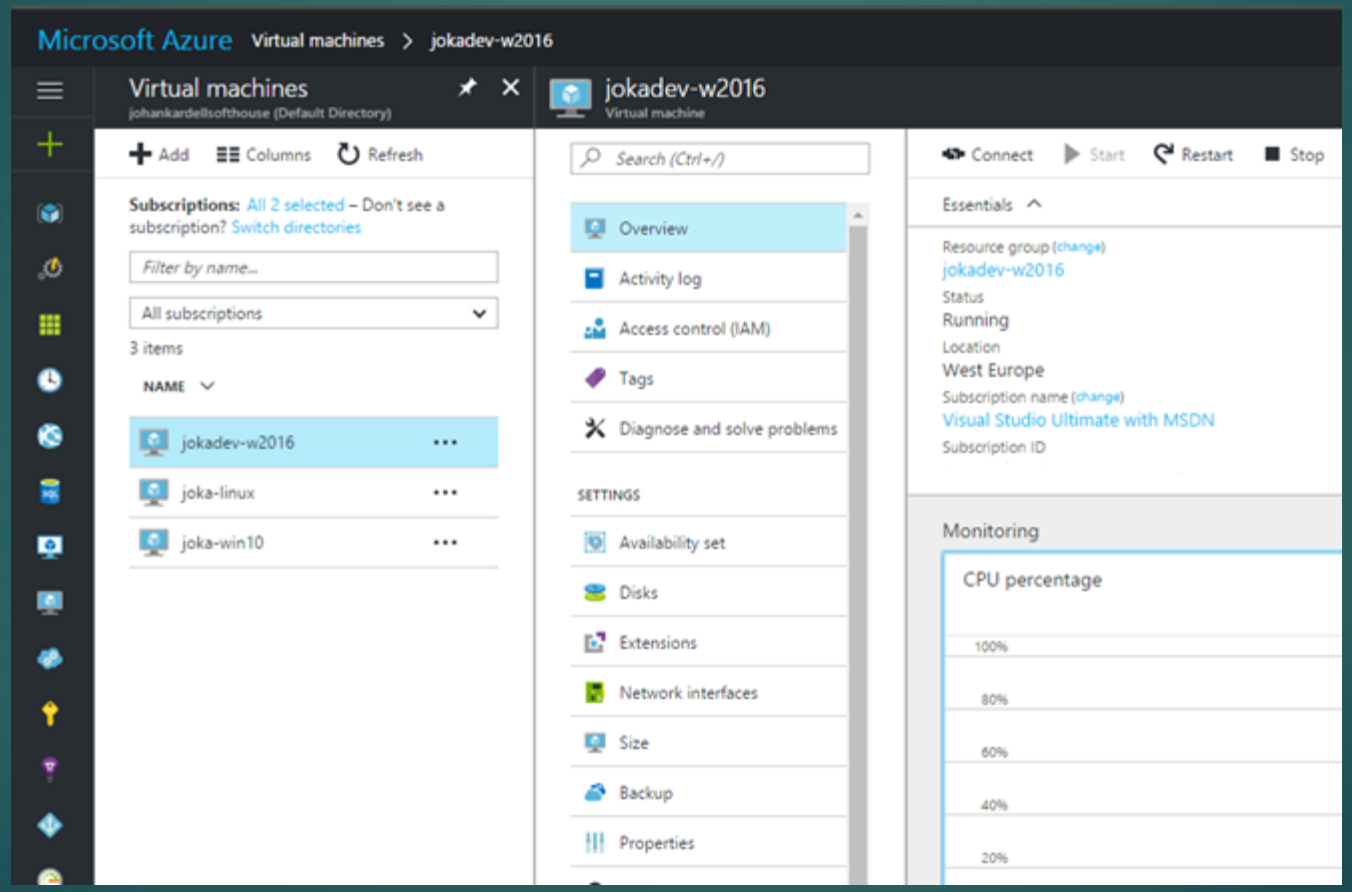
DevOps historical milestones

- 2007: Patrick Debois becomes frustrated by the conflict between Developers and Operations. He ponders solutions.
- 2008 (August): Agile Conference in Toronto, Patrick Debois and Andrew Shafer met during a session called *Agile Infrastructure*.
- 2009 (June): Velocity conference – Flickr famous presentation "*10+ Deploys a Day: Dev and Ops Cooperation at Flickr.*" (Ops who think like devs, devs who think like ops)
- 2009 (October): Patrick Debois coined the term DevOps for DevOpsDays (conference in Belgium)

The concept of Agile infrastructure

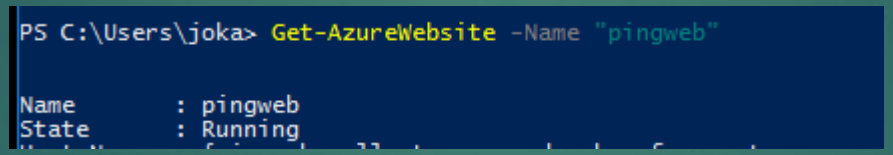
- Infrastructure
  - That supports structured change
  - Allows Traceability
  - Is Testable
  - Can be Automated
  - Continuously Delivered
  - Is described in code

Manual alternative



Azure CLI (Pros)

- The original way of scripting Azure infrastructure setup and config
- Sometimes simple tasks are straight forward and easy to perform



Azure CLI (Cons)

- Traditional CLI (Pipes and parameters).
- Describe actions (Imperative). You have to check state yourself.
- Sometimes simple tasks get complex

- New-AzureVMConfig -Name "MyNewVM" -InstanceSize ExtraSmall -ImageName (Get-AzureVMImage)[4].ImageName ` | Add-AzureProvisioningConfig –Windows –Password $adminPassword ` | New-AzureVM –ServiceName "MySvc2" –AffinityGroup "Contoso"

Azure CLI 2

- Launched feb 2017
- Cleaned up and simplified

- az vm create --name "joka-az-linux" –resource-group "Joka-AZ" --image UbuntuLTS --generate-ssh-keys

Azure Resource Manager

- Microsofts recommended way for Infrastructure as Code in Azure
- **Pros**
  - Declarative. Describe what you want instead of how to do it.
  - Idempotent
  - Simple tasks are kept simple
  - Can be used for complex and large tasks
- **Cons**
  - There are always features not yet available
  - Complex tasks are complex
  - Hard to modularize complex tasks

Azure Resource Manager (Sample)

```
{
    "type": "Microsoft.Web/Sites",
    "apiVersion": "2015-08-01",
    "name": "[parameters('webAppName')]",
    "location": "[resourceGroup().location]",
    "dependsOn": [
        "[concat('Microsoft.Web/serverFarms/', parameters('appSvcPlanName'))]"
    ],
    "properties": {
        "name": "[parameters('webAppName')]",
        "serverFarmId": "[parameters('appSvcPlanName')]",
        "siteConfig": {
```

```
"parameters": {
    "databaseAccountName": {
        "value": "GEN-UNIQUE"
    },
    "appSvcPlanName": {
        "value": "GEN-UNIQUE"
    },
```

Yet even more alternatives

- REST
- C# Code

```
var webApp = azure.WebApps()
    .Define(appName)
    .WithNewResourceGroup(rgName)
    .WithNewAppServicePlan(planName)
    .WithRegion(Region.US_WEST)
    .WithPricingTier(AppServicePricingTier.STANDARD_S1)
    .Create();
```

Conclusion: Gamechangers

- Infrastructure as Code
- Idempotency
  - Scripts no longer contain logic
  - Actions are calculated from description
- Servers are cattle, not pets
  - Recreate servers instead of changing them
  - Infrastructure code is frequently tested
- Build for failure

End-to-end responsibility

- Automation
- Versioning
- Product and infrastructure
  - Built together
  - Tested together
  - Deployed together
  - Monitored together
- Monitoring and surveillance is a team responsibility – no matter where the problem lies