

Azure Web Apps

Azure App Service Family



Web Apps
Web apps that scale
with your business



Mobile Apps
Build mobile apps for
any device



Logic Apps
Automate business
processes across SaaS
and on-premises



API Apps
Build and consume APIs
in the cloud

Azure Web Apps

- Support a variety of languages and platforms
 - .NET, Java, Node.js, PHP, Python, and more
- Support scaling (manual or auto) and load balancing
- Support slots for staged deployments and A/B testing
- Support continuous integration

Familiar and Fast

Leverage existing skills,
plus languages,
frameworks, and tools
you're familiar with

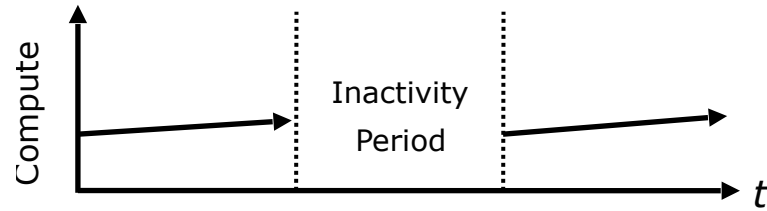
Enterprise Grade

ISO-, SOC2-, and PCO-
compliant with enterprise-
level SLAs

Global Scale

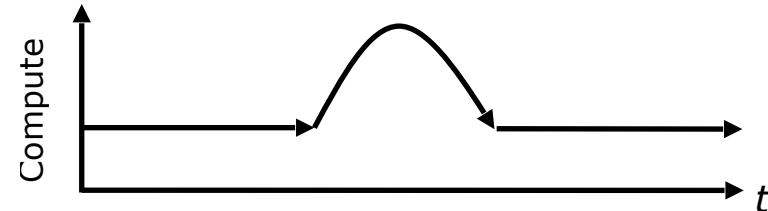
Scale up and down as
needed, manually or
automatically

Scaling - Cloud Computing Patterns



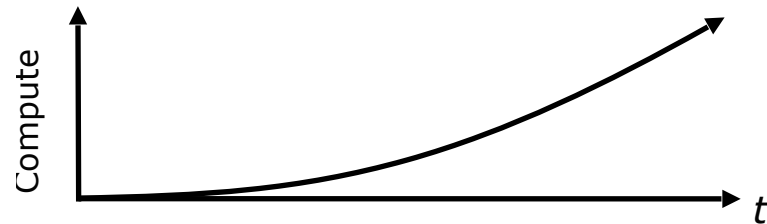
On and Off

On & off workloads (e.g. batch job)
Over provisioned capacity is wasted
Time to market can be cumbersome



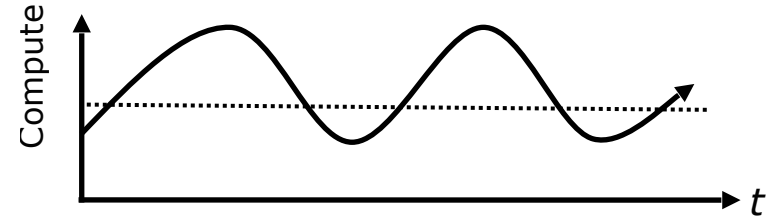
Unpredictable Bursts

Unexpected/unplanned peak in demand
Sudden spike impacts performance
Can't over provision for extreme cases



Growing Fast

Successful services needs to grow/scale
Keeping up w/ growth is big IT challenge
Cannot provision hardware fast enough



Predictable Bursts

Services with micro seasonality trends
Peaks due to periodic increased demand
IT complexity and wasted capacity

Scaling Up vs. Scaling Out

Scale Up



Vary the VM size

*1 Core w/ 1.75 GB RAM
2 Cores w/ 3.5 GB RAM
4 Cores w/ 7 GB RAM*

Scale Out



Vary the VM count

Max 3 instances
Max 10 instances
Max 20/50** instances*


Manual Scaling vs. Auto-Scaling

Manual – Scale via portal or scripts

* Scale by

Description Manual setup means that the number of instances you choose won't change, even if there are changes in load.

Instances




Auto – CPU Percentage

* Scale by

Description Automatically scale up or down based on CPU Percentage. Choose an average value you want to target.

Instances

Target range



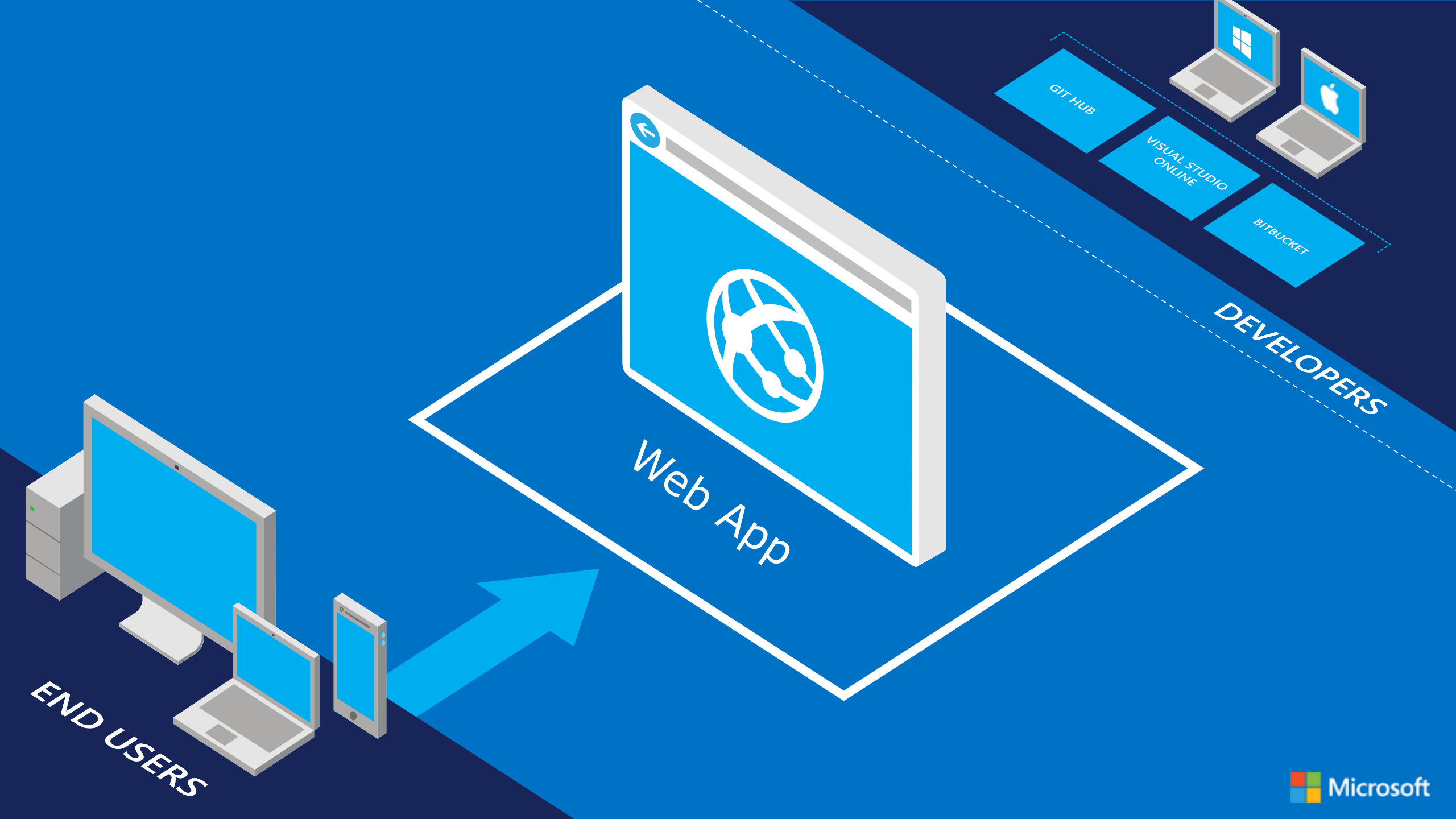
Auto – Schedule & Performance Rules

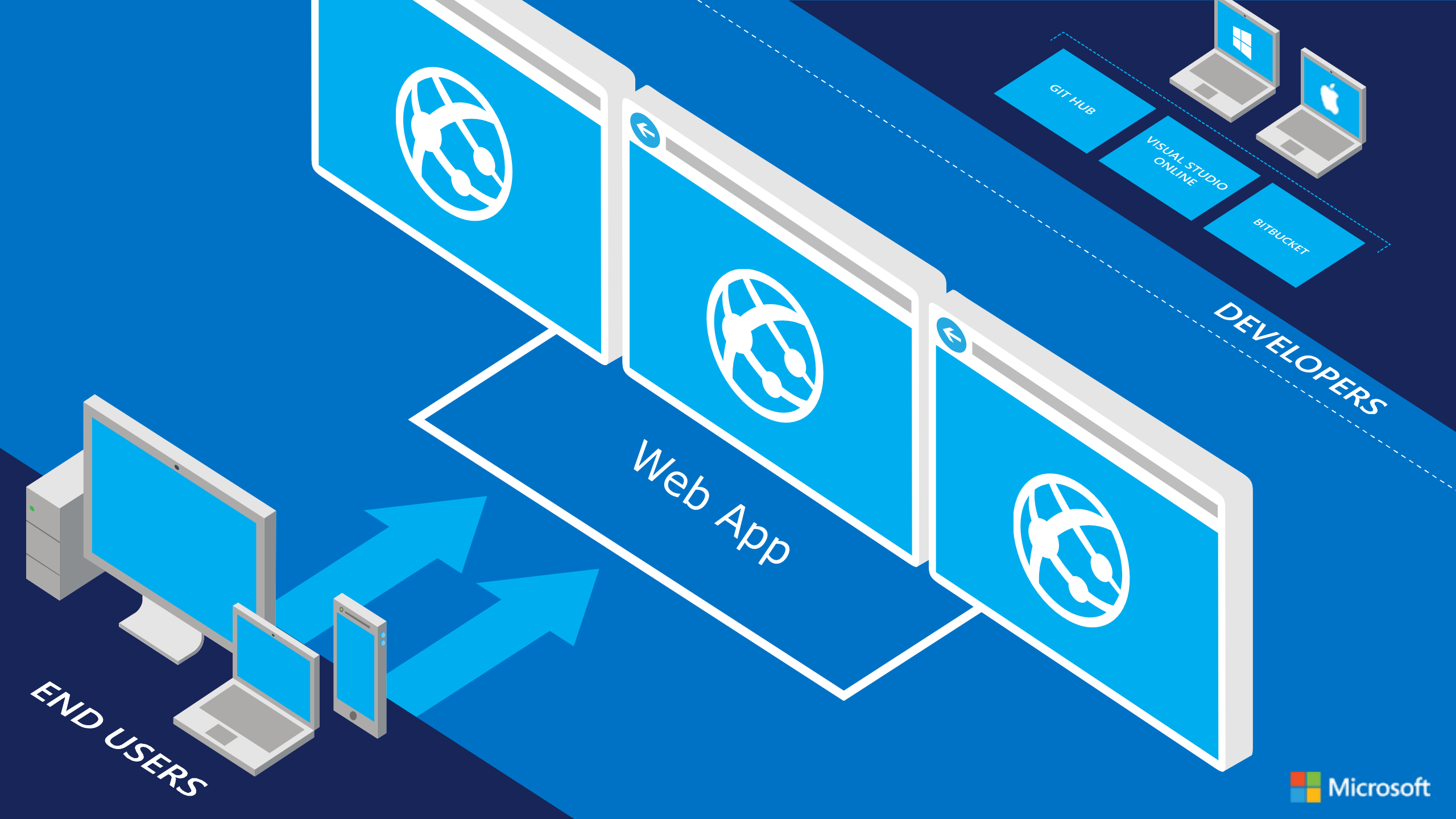
* Scale by

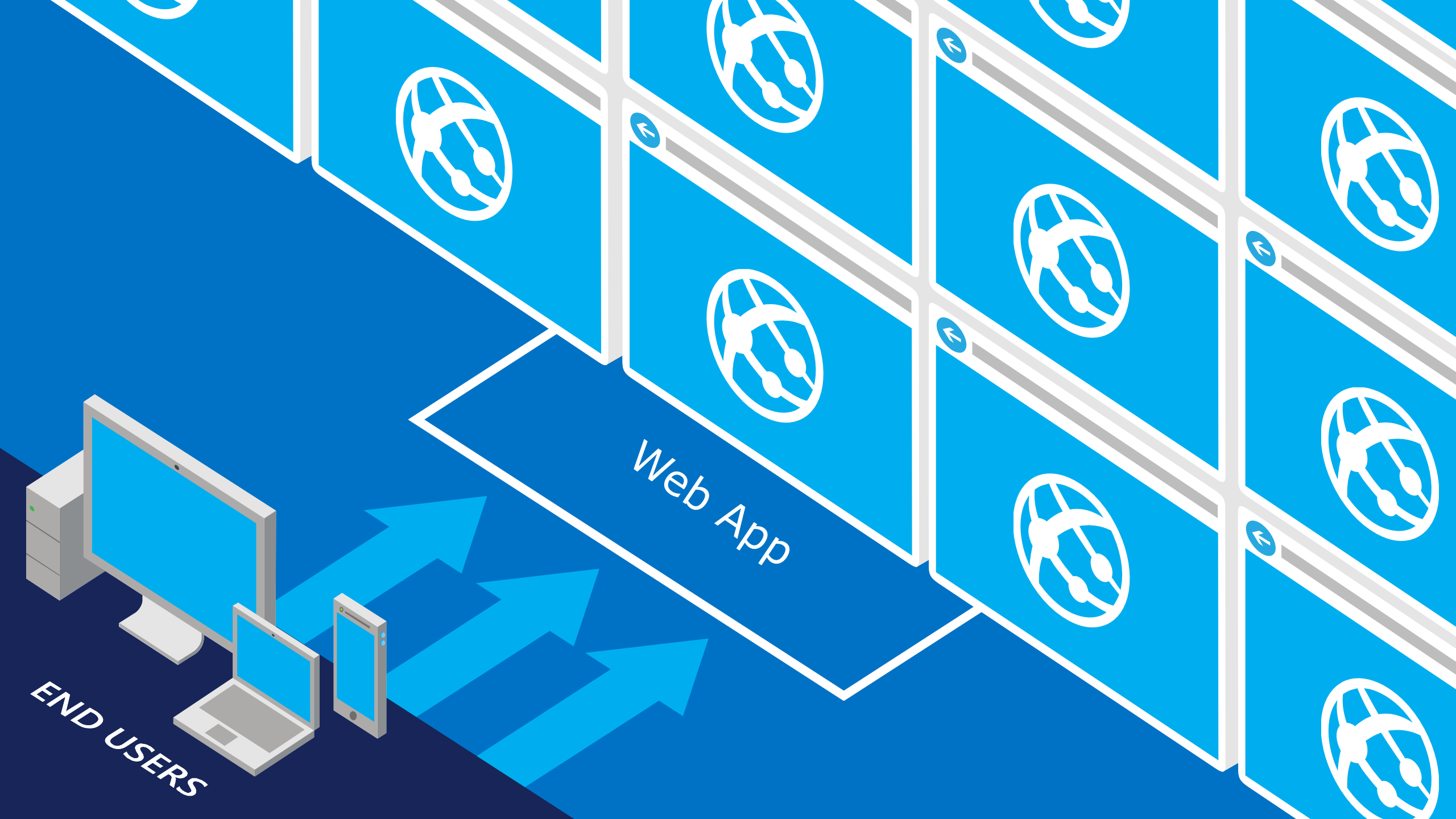
Description Create your own set of rules. Create a schedule that adjusts your instance counts based on time and performance metrics.

Monday-Friday Profile, scale 3 - 9

Settings CPU Percentage > 80 (increase count by 1)

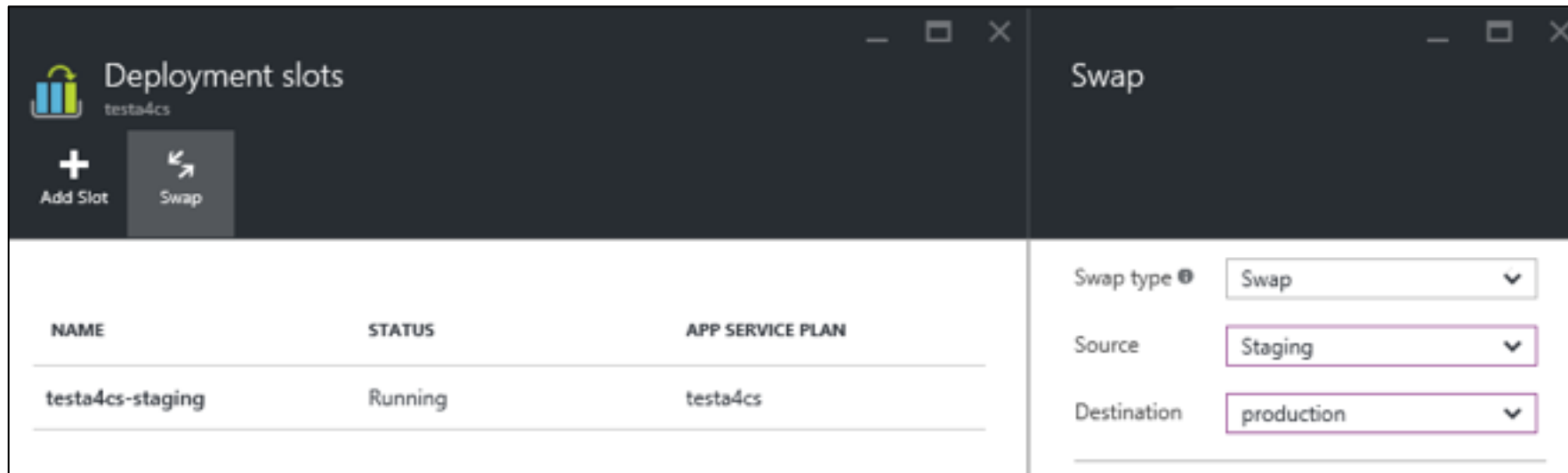






Deployment Slots

- Use a Deploy-Confirm-Promote workflow
 - Promote via “swap” through Azure portal
- <http://sitename-slotname.azurewebsites.net>



The screenshot shows the Azure Portal interface for managing deployment slots for an application named 'testa4cs'. The interface is split into two main panels. The left panel, titled 'Deployment slots', contains a table of existing slots and two buttons: 'Add Slot' and 'Swap'. The right panel, titled 'Swap', contains three dropdown menus for configuring a swap operation.

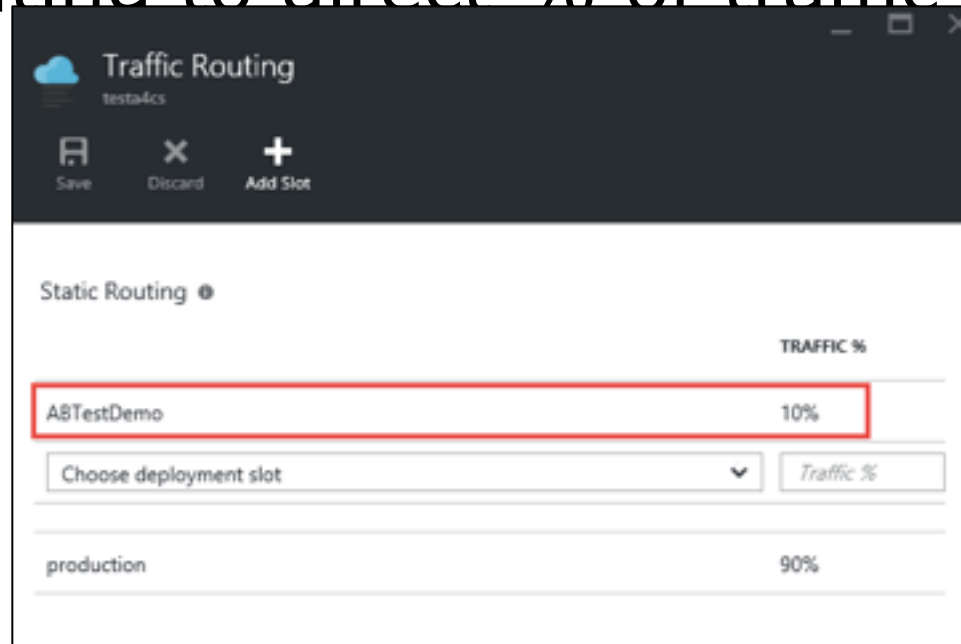
| NAME | STATUS | APP SERVICE PLAN |
|------------------|---------|------------------|
| testa4cs-staging | Running | testa4cs |

Swap configuration:

- Swap type: Swap
- Source: Staging
- Destination: production

A/B Testing

- Test changes by routing requests to different deployment slots
- Use Traffic Routing to direct % of traffic to alternate slots



The screenshot shows the 'Traffic Routing' interface for a resource named 'testa4cs'. It features a dark header with 'Save', 'Discard', and 'Add Slot' buttons. Below, the 'Static Routing' section is active. A table lists routing slots: 'A8TestDemo' with 10% traffic (highlighted with a red box) and 'production' with 90% traffic. Below the table is a 'Choose deployment slot' dropdown menu and a 'Traffic %' input field.

| | TRAFFIC % |
|------------|-----------|
| A8TestDemo | 10% |
| production | 90% |

Continuous Integration

- Web apps can be deployed manually via FTP or WebDeploy
- Automate deployment using 3rd party source-control providers
- Can also use a local Git repository from Azure Portal



Git



Visual Studio



CodePlex



GitHub



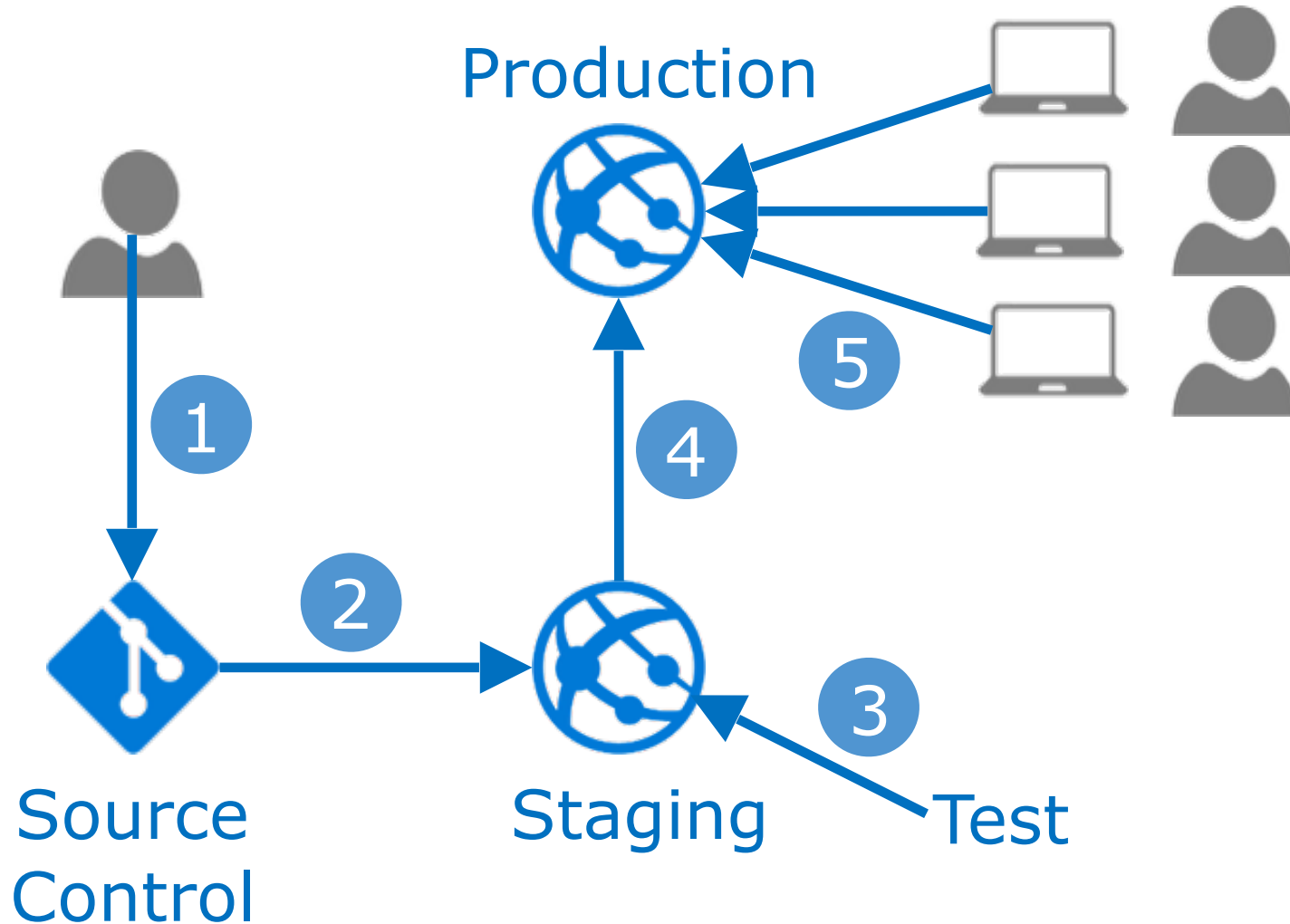
BitBucket



DropBox

Team Services

Continuous Integration + Deployment Slots



1. Developer commits code
2. Automated process builds/compiles and deploys to staging slot
3. Automated and other tests validate content in staging slot
4. Staging content promoted to production
5. Users see updated site