Microsoft Azure

Deploying your Django application for scale with Microsoft Azure

Anthony Shaw, Python Advocate

aka.ms/pycon-django-workshop

Overview

Today we will cover:

- Azure Web Application architecture
- How to deploy Django on Azure
- Tips and best practices for Django on Azure
- How to scale a Django application from tens to thousands of users
- How to build the right architecture for millions of users
- Managing and monitoring real-world Django applications

aka.ms/pycon-django-workshop

Microsoft Azure

Outcomes from this talk

After watching this talk, you will know

- What services are available to run a full Django application
- Cloud Architecture for Django and how it applies to Azure
- How to deploy a Django application on Azure
- How to scale applications up and out depending on your workload
- How to setup monitoring and dashboards

aka.ms/pycon-django-workshop

Expectations

The expectations of this talk are that you already know:

- Basic Python programming
- How to use a code editor
- What Django is and the basic concepts of Django

We also assume you have an application on Django that you've made or are making.

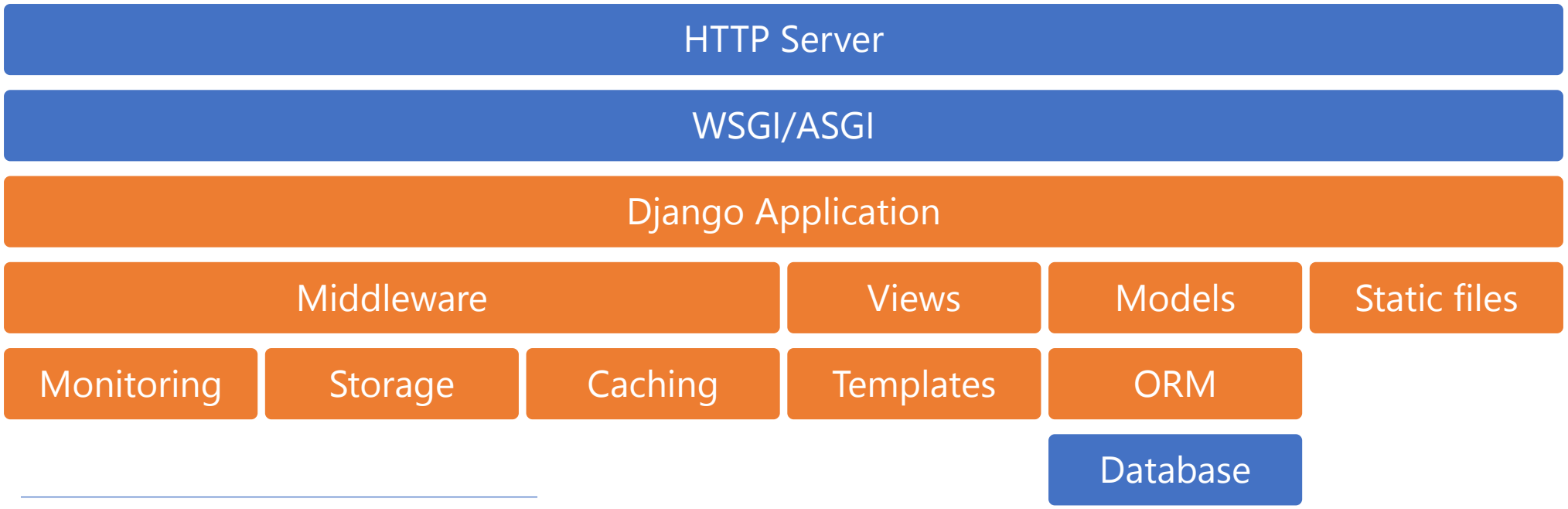
If you need a Django tutorial, we have some links at the end.

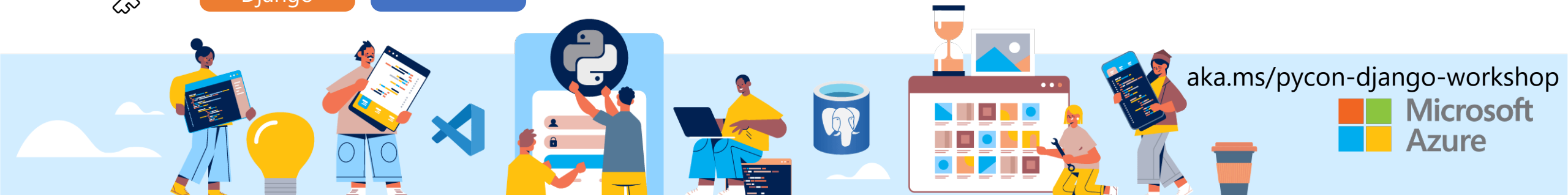
1. Architecture

Microsoft Azure

aka.ms/pycon-django-workshop

Review of Django Architecture

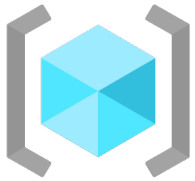




Basic Concepts

Subscription

A subscription is your Azure account.

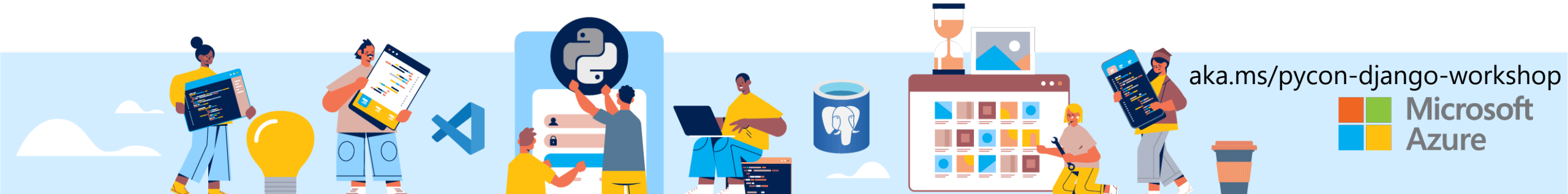


Resource Group

A resource group is a container (like a folder) to hold all the resources required by your app. Resource groups are useful to prevent your subscription getting cluttered and confusing when you have lots of applications

Runtime environment

Azure offers plenty of ways to run Python applications. App Services and VMs are the most used.



aka.ms/pycon-django-workshop

Microsoft Azure

Choosing the right architecture

App Services

- Linux Virtual Machines
- Supports Python 3.6-3.8
- Scale-out options available
- Less maintenance overheads
- Usage based pricing

Virtual Machines

- Linux and Windows VMs available
- You are responsible for patching and securing the servers
- Suitable for non-standard system requirements

aka.ms/pycon-django-workshop

Microsoft Azure

2. Azure Web Apps

Microsoft Azure

aka.ms/pycon-django-workshop

App Services Components

Web Apps

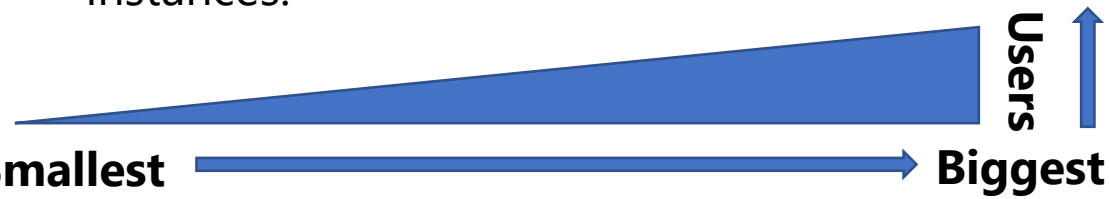
The container that holds your application, its configuration, certificates and deployment state.

App Service Plans

The plan that controls how many slots (containers) the code is deployed onto, and what specification they are.

Instances

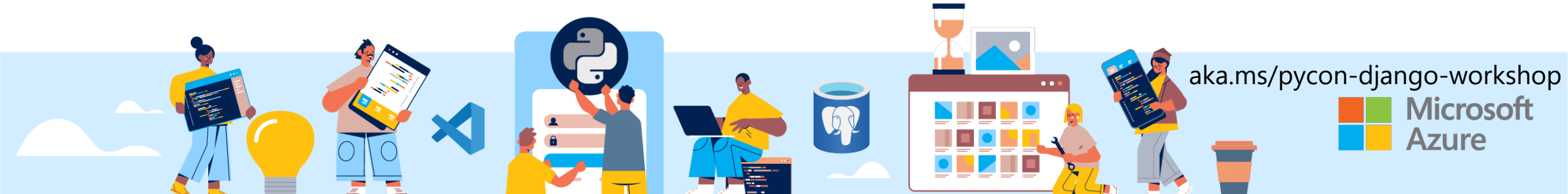
Instances are containers with your application deployed. You can have 1-30 instances.



B1
- 1.75GB RAM
- 1 vCPU

...10 options in between

P3v3
- Free
- 32GB RAM
- 8 vCPU



aka.ms/pycon-django-workshop

Microsoft Azure

Example Architectures

Simple Dev/Test Environment

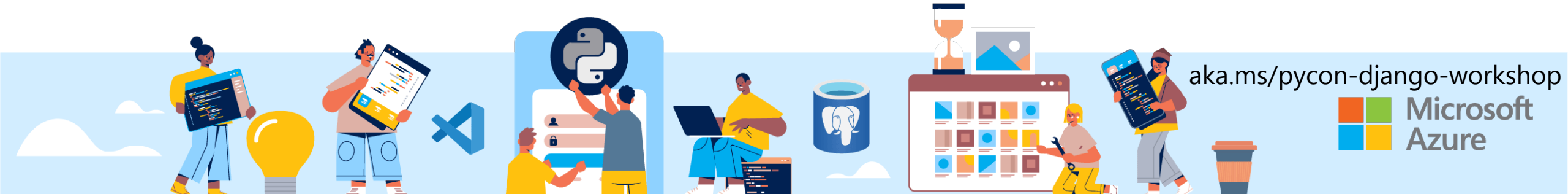
- Web App
- Dev/Test Plan
- 1x B1 Instance

Small Production Environment

- Web App
- Production Plan
- 1x P1v2 Instance

Medium Production Environment

- Web App
- Production Plan
- 4x P2v2 Instances



aka.ms/pycon-django-workshop

Microsoft Azure

Tips to scaling your web applications

Microsoft Azure

- ✓Experiment with different sizes
- ✓Run a load-test using Locust IO
- ✓Favor a cluster over a single, massive instance
- ✓Move as much of the load off the app as you can by:
  - ❖ Template caching
  - ❖ Static files on CDN

aka.ms/pycon-django-workshop

Configuring an ASGI worker

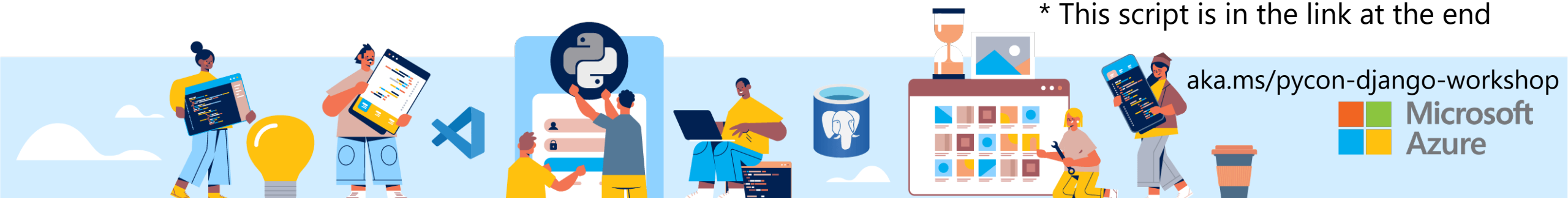
Tip

Django 3 supports asynchronous web workers. Even for a Django application that doesn't have async views, configuring ASGI with uvicorn can make your application faster.

1. Add the following startup.sh script
2. Make sure you add uvicorn to the requirements.txt file
3. Pick the right number of workers and threads for the instance size
4. To enable this startup command, you need to set the startup command to startup.sh in Settings -> Configuration -> General Settings -> Startup command. After making these changes, the application will restart.

```
gunicorn --workers 8 --threads 4 --timeout
60 --access-logfile '-' --error-logfile '-'
--bind=0.0.0.0:8000 -k
uvicorn.workers.UvicornWorker --
chdir=/home/site/wwwroot
your_django_app.asgi
```

* This script is in the link at the end



aka.ms/pycon-django-workshop

Microsoft Azure

3. Databases

Microsoft Azure

aka.ms/pycon-django-workshop

Azure Database Services

Azure Database for Postgres

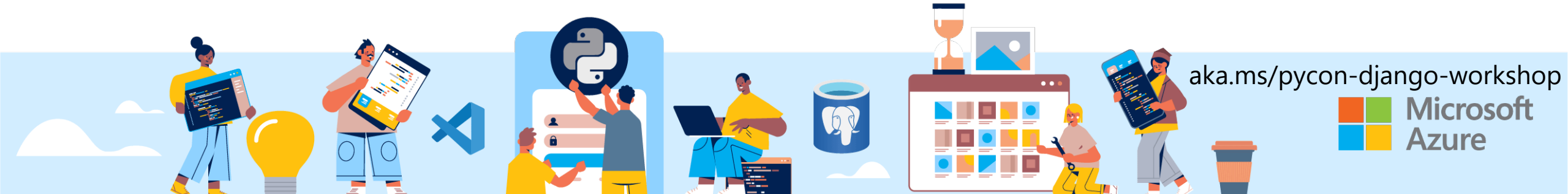
A managed PostgreSQL database-as-a-service. Microsoft runs Postgres for you. Fully compatible with Django.

Azure SQL

A managed Microsoft SQL Server database-as-a-service. Compatible with Django ORM through the mssql ORM driver.

Azure Database for MySQL/MariaDB

Both MySQL and MariaDB are available in database-as-a-service. Fully compatible with Django.



aka.ms/pycon-django-workshop

Microsoft Azure

Django Database Support

Django officially supports:

- PostgreSQL
- MariaDB
- MySQL
- Oracle**
- SQLite**

3rd party extensions support:

- Microsoft SQL Server
- Cockroach DB**
- MongoDB**
- Firebird**

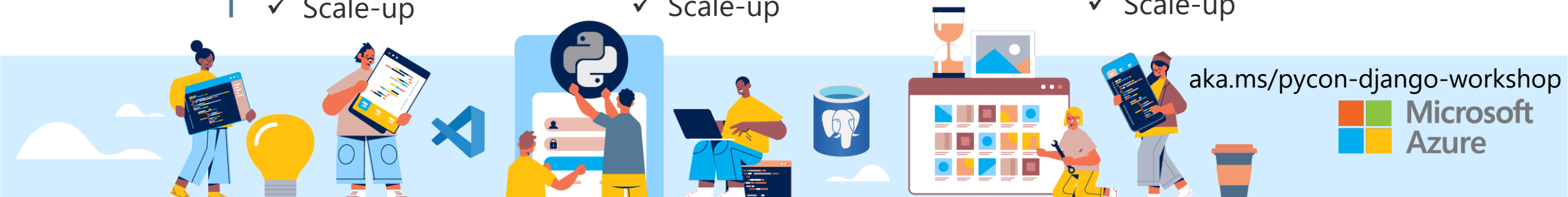
** can be deployed on Azure using a VM, no database-as-service offering available

Microsoft Azure

aka.ms/pycon-django-workshop

Azure Database for PostgreSQL options

	Lightweight	General	Memory Intensive
Single	Basic Single server ✓ 1 - 64 vCores ✓ 5GB – 1TB storage ✓ Local redundancy ✓ Scale-up	General Single server ✓ 2 - 64 vCores ✓ 5GB – 15TB storage ✓ Local redundancy ✓ Geo redundancy ✓ Scale-up	Memory Optimized ✓ 2 - 64 vCores ✓ 5GB – 15TB storage ✓ Local redundancy ✓ Geo redundancy ✓ Scale-up
Flexible (Preview)	Burstable ✓ 1 - 2 vCores ✓ 32GB – 15TB storage ✓ 2-4 GiB Memory ✓ Local redundancy ✓ Scale-up	General Purpose ✓ 2 - 64 vCores ✓ 32GB – 15TB storage ✓ 8-256 GiB Memory ✓ Local redundancy ✓ Scale-up	Memory Optimized ✓ 2 - 64 vCores ✓ 32GB – 15TB storage ✓ 16-432 GiB Memory ✓ Local redundancy ✓ Scale-up



aka.ms/pycon-django-workshop

Microsoft Azure

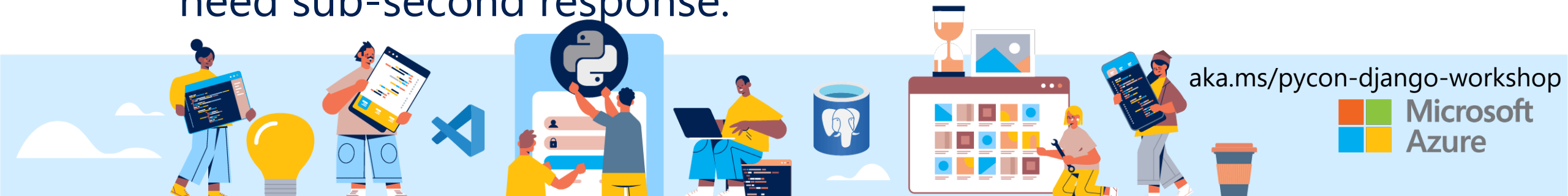
Scale-Out Options for PostgreSQL

Hyperscale (Citus) server group

- Best for ultra-high performance and data needs beyond 100GB.
- Ideal for multi-tenant applications and real-time analytical workloads that need sub-second response.

Azure Arc enabled PostgreSQL Hyperscale (Preview)

- Best for ultra-high performance and data needs beyond 100GB **on your infrastructure.**



aka.ms/pycon-django-workshop

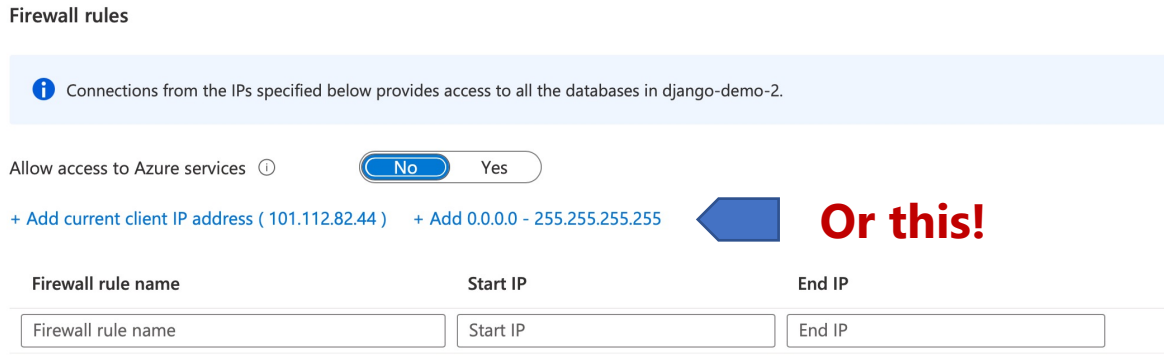
Microsoft Azure

Securing application to database traffic

Tip

Because you can end up with multiple instances of your app running and the IP addresses will change, setup a Network Security Group between PostgreSQL and the App Service Plan to control PostgreSQL security.

Don't do this!



Or this!

Microsoft Azure

aka.ms/pycon-django-workshop

4. Content Delivery

Microsoft Azure

aka.ms/pycon-django-workshop

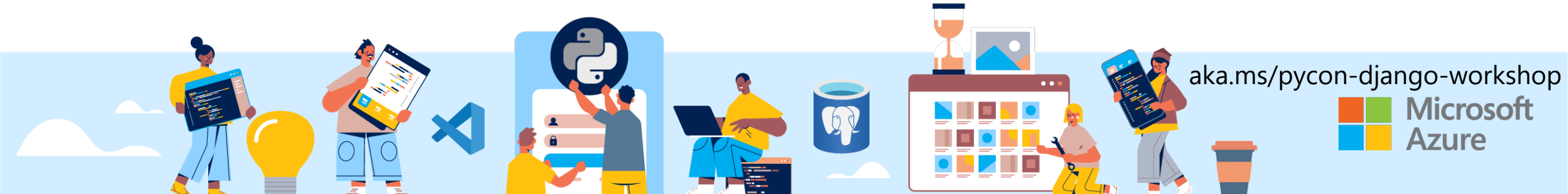
Configuring Django static files with Azure

Tip

Moving static files off your application instances and onto a CDN is a great way to improve performance **and** reduce cost.

1. Install **django-storages[azure]** into your virtual environment and add it to your requirements.txt file
2. Add **'storages'** to the list of **INSTALLED_APPS**
3. Create a backend module inside your application and define two (or more) classes for static and media files

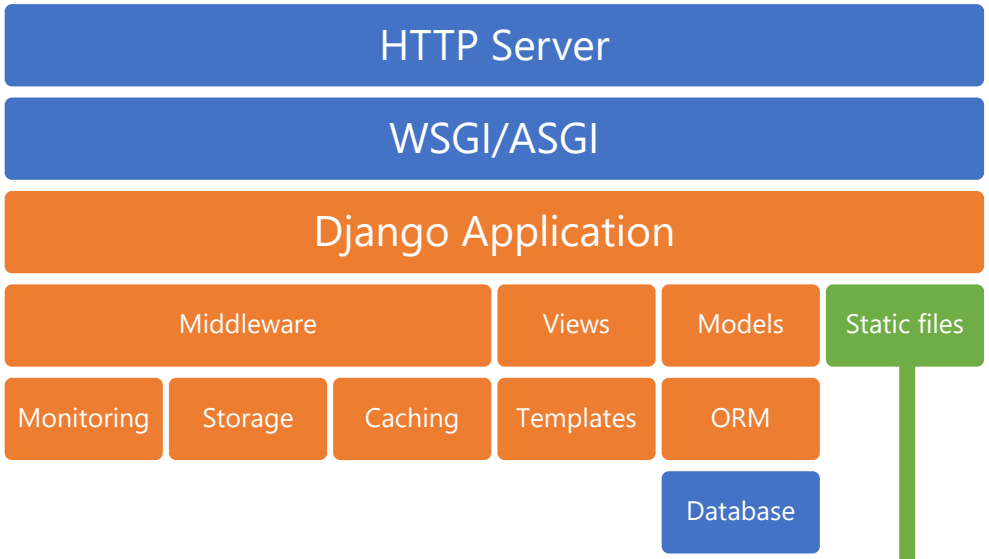
There are plenty of options for the *AzureStorage* class, check them out at the plugin documentation.



aka.ms/pycon-django-workshop

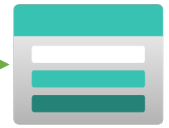
Microsoft Azure

Offloading static files to Azure



Azure CDN Caching Endpoint (Optional)

- Replicates your content to datacenters across the world
- Dramatically reduces load times



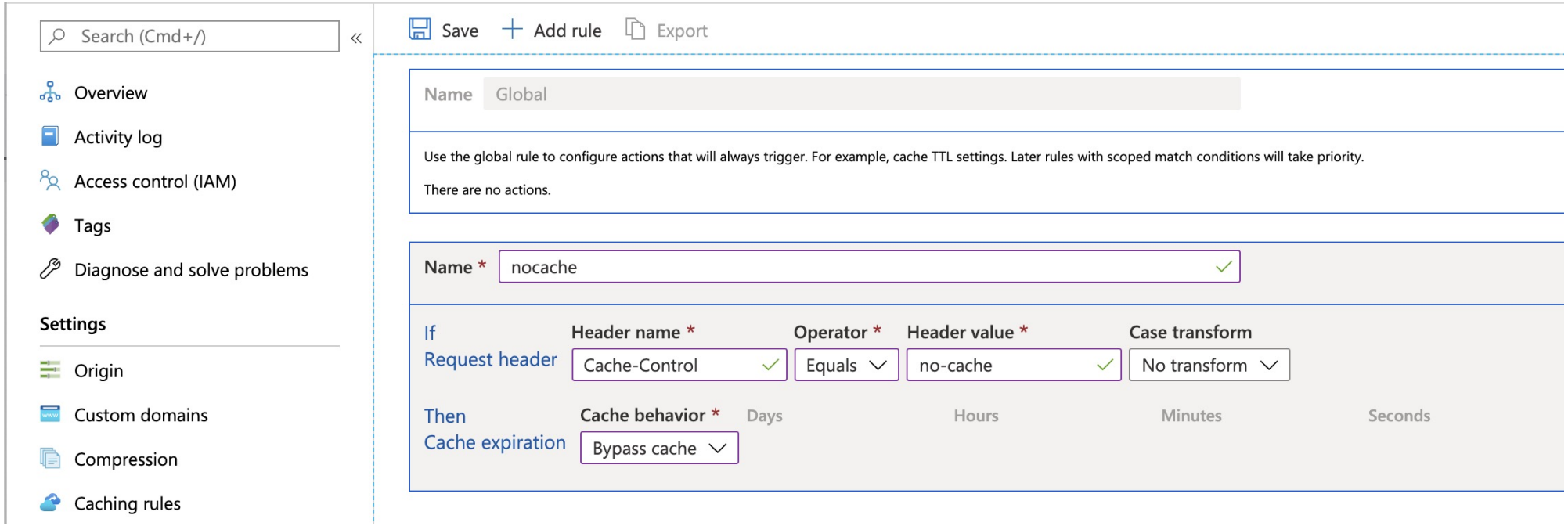
Azure Storage Blob

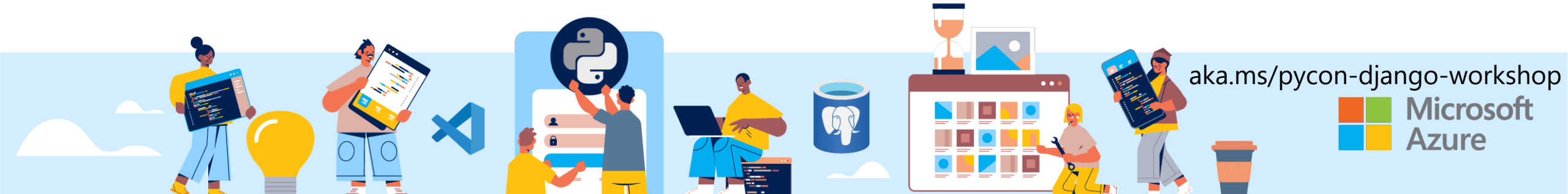
Microsoft Azure

aka.ms/pycon-django-workshop

Tip

Add a custom rule in Azure CDN to bypass cache when the "Cache-Control : no-cache" header is present. This means you can force the CDN to refresh from your browser by doing a hard refresh.





aka.ms/pycon-django-workshop

Microsoft Azure

5. Monitoring and Insights

Microsoft Azure

aka.ms/pycon-django-workshop

Configuring Application Insights

Django Middleware

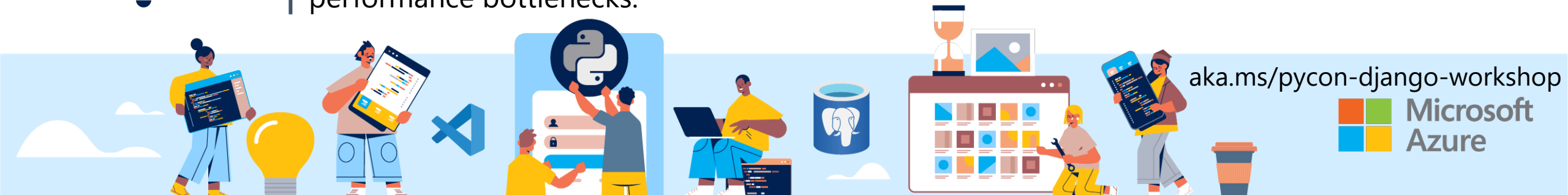
Use the opencensus middleware for Django with the Azure exporter to capture all your web requests, error and trace data.

Application Insights

Application insights will aggregate all requests across all instances and allow you to explore response times, look at tracebacks of errors and see user flows

Tip

Include the Application Insights extension for JavaScript in your Django main template to capture client-side information, like unhandled JavaScript exceptions, load times and performance bottlenecks.



aka.ms/pycon-django-workshop

Microsoft Azure

Setting up a Dashboard

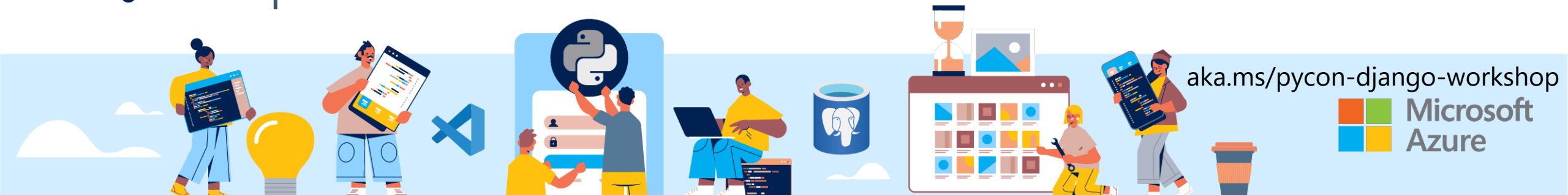
All the Azure services mentioned so far have performance data. You can collect this data in a dashboard to see an overall view of the infrastructure performance and load.

I recommend the following:

- App Plan - CPU Utilization
- App Plan - Memory Utilization
- Web App - Requests
- Web App - Response Time
- Database - Resource Utilization

Tip

The Azure Mobile app is a great way to find resources, or look at the status of your application on the run.



aka.ms/pycon-django-workshop

Microsoft Azure

6. Deployment and DevOps

Microsoft Azure

aka.ms/pycon-django-workshop

Deployment Options

Manual Deployment

To get started, you can deploy from:

1. VS Code using the Azure Extension
2. The Azure Portal by connecting it to a secure FTP server
3. The Azure CLI

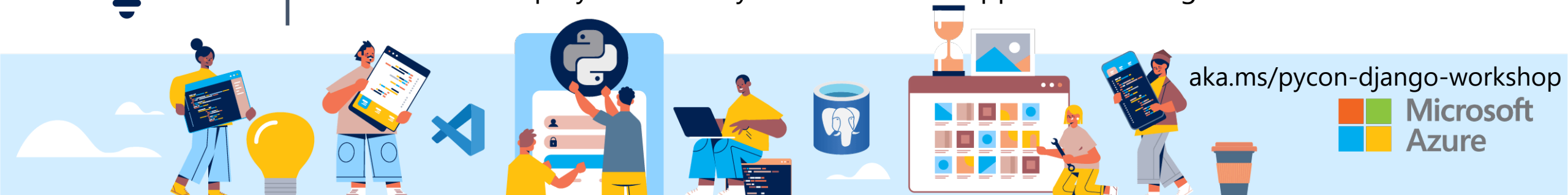
Automated Deployment

There are plenty of options, depending on your requirements:

1. GitHub Actions, for projects hosted on GitHub.com
2. Azure DevOps/Pipelines for projects hosted on private Git Servers
3. Azure CLI for projects hosted on completely private infrastructure

Tip

Start off with something simple, like a manual deployment from VS Code. You can move to an automated deployment once you've tested the application configuration.



7. Extra Components

Microsoft Azure

aka.ms/pycon-django-workshop

Other services you might need

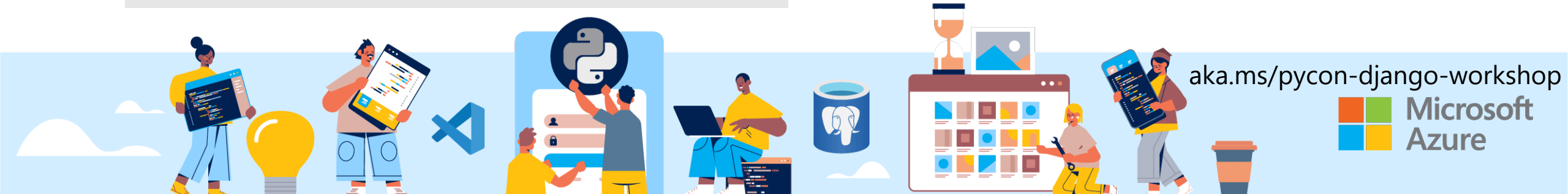
Mail services

- Azure offers a SMTP mail service through a partner, Sendgrid.
- Sendgrid is configurable with as Django middleware.

```
SENDGRID_API_KEY =
os.getenv('SENDGRID_API_KEY’)
EMAIL_HOST = 'smtp.sendgrid.net’
EMAIL_HOST_USER = 'apikey’ # Yes, that’s it.
EMAIL_HOST_PASSWORD = SENDGRID_API_KEY
EMAIL_PORT = 587 EMAIL_USE_TLS = True
```

Celery replacement

- Azure Functions are a great place to run short, scheduled utilities.
- Use a Timer Trigger or HTTP Trigger for your functions.



8. Summary

Microsoft Azure

The whole picture

App Services

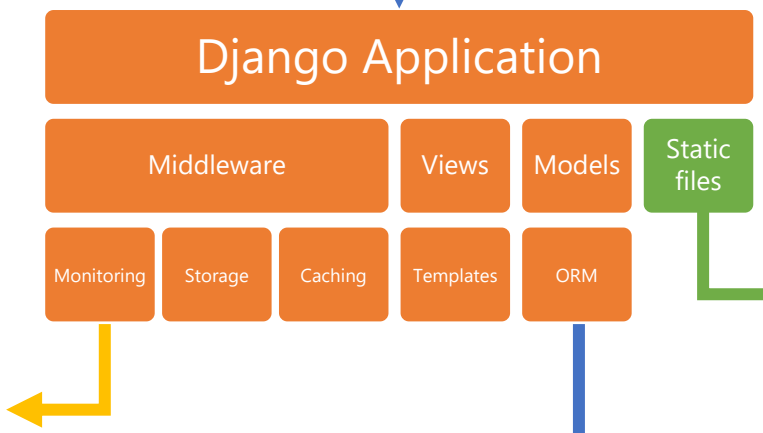
Orchestrates the deployment of your code onto instances and handles scaling

Application Insights

Collects and archives all the performance and error data from your application

Application Instances

Serves the application HTTP service and executes your Django app



Azure CDN

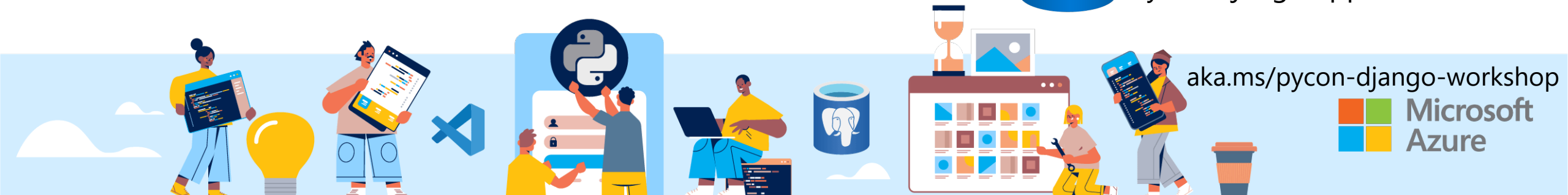
Replicates and delivers your static files globally

Azure Storage

Stores your media and static files

Azure Database for Postgres

Scalable Database service for your Django application



aka.ms/pycon-django-workshop

Microsoft Azure

Things to remember

- ✓Don’t overprovision, run a load test before committing to an architecture
- ✓Experiment with moving components like static files off your application servers
- ✓Cache, cache, cache!
- ✓Test your monitoring and error-handling setup

Conclusion and questions

A copy of these slides as well as resources and links for everything shown today can be found at --

https://aka.ms/pycon-django-workshop