



AZ-204 Compute 1: Azure App Service Web Apps

AZ-204 domain 1, Develop Azure compute solutions | Plans and tiers, deploy, configuration, deployment slots and swap, autoscale, TLS, diagnostics logs; az CLI; as of 2026

AZ-204 Domain 1 App Service Slots Autoscale

App Service in one view

Azure App Service is the managed PaaS (platform as a service) for HTTP apps: web apps, REST APIs and mobile back ends on Windows or Linux, from code (.NET, Java, Node.js, Python, PHP) or a container image. The platform patches the OS and runtime, terminates TLS, load-balances instances and offers slots, autoscale and built-in authentication (Easy Auth). Apps run on an **App Service plan**: every app in a plan shares its instances, so a noisy app slows its neighbours.

Tier	Compute	Key features	Max instances
Free F1 / Shared D1	shared VMs, CPU minutes per day	dev only; no custom TLS, no slots, no scale-out, no Always On	1
Basic B1-B3	dedicated	custom domains + TLS, manual scale, Always On, VNet integration, private endpoints; no slots, no autoscale	3
Standard S1-S3	dedicated	+ autoscale (rules), 5 slots, daily backups, Traffic Manager	10
Premium v3 / v4 (P0v3-P5mv3, P*v4)	dedicated, faster CPUs, memory-optimised (m)	+ 20 slots, automatic scaling, zone redundancy, more storage	30
Isolated v2 (I1v2-I6v2)	dedicated, inside an App Service Environment v3 (ASEv3) in your VNet	single-tenant, internal load balancer option, highest scale and isolation	100

Plan tiers (as of 2026; check the pricing page for current SKUs and limits)

EXAM TIP

Keyword → tier: 'deployment slots' or 'autoscale' → **Standard** minimum; 'more than 5 slots', 'zone redundant', 'automatic scaling' → **Premium v3**; 'network-isolated, single tenant, private only' → **Isolated v2 (ASE v3)**; 'custom domain with TLS, cheapest' → **Basic**. Scaling **up** = change tier/size (more CPU/RAM per instance); scaling **out** = more instances.

Create and deploy

Build the create-app command

- 0 `$ az_webapp_create`
- 1 `$ az webapp create -g rg-shop -n shop-api-prod`
resource group and globally unique name (hostname prefix)
- 2 `$ az webapp create -g rg-shop -n shop-api-prod -p plan-shop-p1v3`
existing plan: `az appservice plan create --sku P1V3 --is-linux`
- 3 `$ az webapp create -g rg-shop -n shop-api-prod -p plan-shop-p1v3 --runtime "PYTHON:3.12"`
list values with `az webapp list-runtimes --os linux`
- 4 `$ az webapp create -g rg-shop -n shop-api-prod -p plan-shop-p1v3 --runtime "PYTHON:3.12" --assign-identity [system]`
system-assigned managed identity for Key Vault and storage
- 5 `$ az webapp create -g rg-shop -n shop-api-prod -p plan-shop-p1v3 --runtime "PYTHON:3.12" --assign-identity [system] --https-only true`
redirect HTTP to HTTPS

Command	What it does
<code>az webapp up --runtime PYTHON:3.12 --sku B1</code>	one-shot: create group, plan and app from the current folder, then zip-deploy it
<code>az webapp deploy --src-path app.jar --type jar</code>	Kudu OneDeploy API: zip, war, jar, ear, static or startup files
<code>az webapp config container set --container-image-name acr.azurecr.io/api:1.4</code>	Web App for Containers: point the app at an image
<code>az webapp deployment source config-local-git</code>	push-to-deploy Git remote on the SCM site
<code>az webapp log tail -n shop-api-prod</code>	live log stream (stdout/stderr and application logs)

- **Deployment paths** zip/OneDeploy (CI artefact), run from package, GitHub Actions (azure/webapps-deploy), Azure Pipelines (AzureWebApp@1), container image from ACR (with continuous deployment webhook), local Git; disable FTP and basic-auth publishing credentials in production
- **Containers** WEBSITES_PORT tells App Service the port the container listens on; pull from ACR with the managed identity (acrUseManagedIdentityCreds) instead of

admin credentials; /home is the persistent share (WEBSITES_ENABLE_APP_SERVICE_STORAGE)

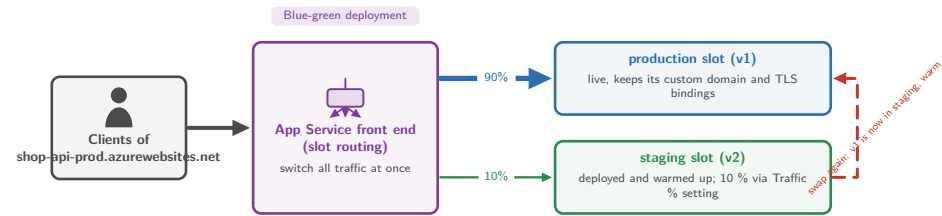
☀ Configuration: settings, secrets, TLS

App settings	exposed as environment variables; encrypted at rest; overrides appsettings.json / Web.config keys; __ (double underscore) maps to : for nested .NET keys on Linux
Connection strings	env vars with a type prefix: SQLAZURECONNSTR_, SQLCONNSTR_, MYSQLCONNSTR_, POSTGRESQLCONNSTR_, CUSTOMCONNSTR_ (+ name); .NET reads them from ConnectionStrings
Key Vault reference	@Microsoft.KeyVault(SecretUri=https://kv-shop.vault.azure.net/secrets/DbPwd/) or (VaultName=kv-shop;SecretName=DbPwd); needs a managed identity with Key Vault Secrets User ; omit the version to pick up rotations (refetched within 24 h or on restart)
General settings	stack and version, Always On (Basic+, keeps the app loaded), ARR affinity, HTTP version, 32/64-bit, FTP state, minimum TLS 1.2 (default), client certificates
TLS certificates	free App Service managed certificate (auto-renewed, no wild-card, not exportable), App Service certificate (purchased), import from Key Vault, upload PFX; bind with SNI SSL or IP SSL
Restarts	changing app settings or connection strings restarts the app; a red Key Vault reference status usually means a missing role assignment or a vault firewall

```
Set settings, one of them sticky to the slot
az webapp config appsettings set -g rg-shop -n shop-api-prod \
--settings FEATURE_X=on \
"DbPwd=@Microsoft.KeyVault(SecretUri=https://kv-shop.vault.azure.net/secrets/DbPwd/)"
az webapp config appsettings set -g rg-shop -n shop-api-prod \
--slot staging --slot-settings ENV_NAME=staging
az webapp config connection-string set -g rg-shop -n shop-api-prod \
-t SQLAzure --settings Orders="Server=tcp:..."
```

🔗 Deployment slots and swap

Staging slot swap, optionally with a canary split



- 1 **Apply target settings to the source** production's slot-specific settings and connection strings, plus its continuous-deployment and auth settings, are applied to every staging instance (staging restarts)
- 2 **Warm up** each staging instance gets an HTTP request on / (or applicationInitialization paths, WEBSITE_SWAP_WARMUP_PING_PATH, accepted codes in WEBSITE_SWAP_WARMUP_PING_STATUSES); any failure cancels the swap
- 3 **Switch routing** the front end swaps host names: warmed instances now serve production with no dropped requests
- 4 **Finish the old production** the former production content gets staging's settings and restarts, ready for a swap back (rollback)

What a swap of staging into production does

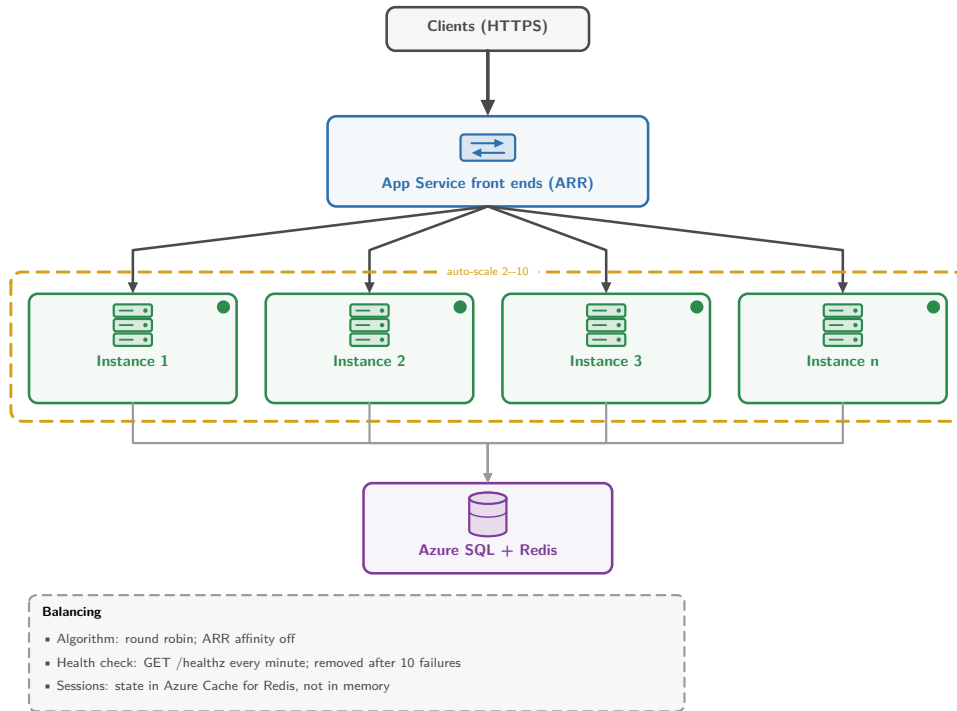
Swapped	Stays with the slot (never swapped)
general settings (stack, version, 32/64-bit, web sockets)	publishing endpoints and credentials
app settings and connection strings not marked as slot setting	custom domain names, non-public certificates, TLS bindings
handler mappings, public certificates	scale settings, Always On, IP restrictions, CORS
WebJobs content, hybrid connections	WebJobs schedulers, diagnostic settings, VNet integration
path mappings (Windows)	managed identities, settings ending in _EXTENSION_VERSION
Swapped with the content vs sticky to the slot	
az webapp deployment slot create -n shop-api-prod -g rg-shop --slot staging --configuration-source shop-api-prod	create a slot, cloning production's configuration (content is not cloned)
az webapp deployment slot swap -n shop-api-prod -g rg-shop --slot staging --target-slot production	swap; add --action preview then --action swap (or reset) for swap with preview
az webapp traffic-routing set -n shop-api-prod -g rg-shop --distribution staging=10	send 10 % of production traffic to staging; the x-ms-routing-name cookie pins the client

⚠ WARNING

A secret that must differ between environments (staging database, staging API key) has to be a **slot setting**, otherwise the swap carries the staging value into production. Testers reach a slot directly with ?x-ms-routing-name=staging; x-ms-routing-name=self returns to production.

Scale out and stay healthy

One plan, autoscaled, stateless instances



Autoscale (Azure Monitor rules)

Standard tier and above

you write rules: metric, operator, threshold, time grain, statistic, duration, action, cool-down

profiles: default, recurrence (weekday 08-18), fixed date (Black Friday)

scale out and in rules must not flap: in-threshold well below out-threshold

Automatic scaling (Premium)

Premium v2 / v3 / v4 plans

platform decides from HTTP load; set maximum burst and always-ready (pre-warmed) instances per app

no rules; works per app inside the plan

pre-warmed instances absorb bursts with no cold start

PROBLEM Size the autoscale rules for an API on S2 instances

- GIVEN**
- one S2 instance serves 250 requests per second at 70 % CPU
 - peak 2,000 requests per second, night trough 300
 - Standard tier maximum 10 instances

1 Instances at peak

$$N = \text{ceil}(2,000 / 250) = 8$$

Why: $N = \text{peak rate} / \text{per-instance rate at the target CPU}$

2 Check the tier limit

$$8 \leq 10 \text{ on Standard}$$

Why: above 10 you must move to Premium v3 (30)

3 Minimum

$$\text{ceil}(300 / 250) = 2, \text{ and } 2 \text{ for resilience}$$

Why: never 1 instance in production: a platform upgrade would cause downtime

4 Rules

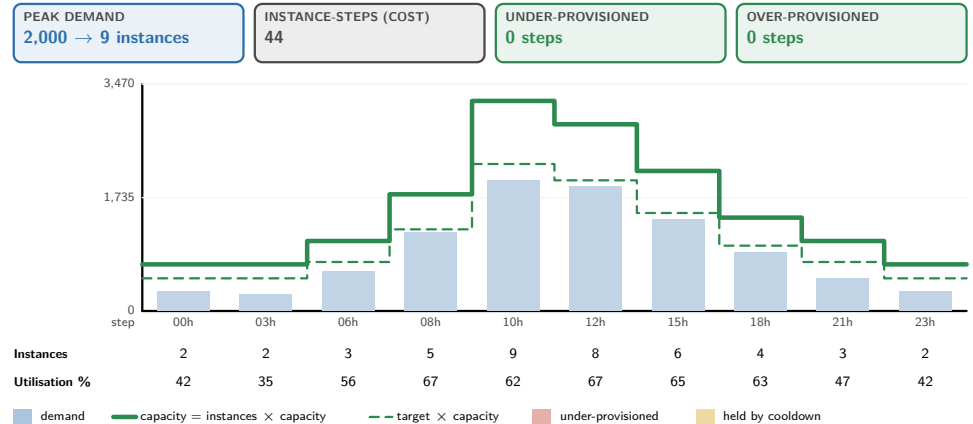
out: avg CPU > 70 % over 10 min, +2, cool-down 5 min; in: avg CPU < 35 % over 10 min, -1, cool-down 10 min

Why: a gap between thresholds and a slower scale-in stop flapping

ANSWER Autoscale profile min 2, max 10, default 2; scale out by 2 above 70 % CPU, in by 1 below 35 %

CHECK ✓ 8 instances x 250 req/s = 2,000 req/s at peak

The same API over a day: target 70 % CPU, min 2, max 10



Wanted instances = $\lceil \text{demand} \div (\text{capacity per instance} \times \text{target}) \rceil = \lceil \text{demand} \div (357 \times 0.7) \rceil$, kept between min 2 and max 10; utilisation = $\text{demand} \div (\text{instances} \times \text{capacity})$. Scale-out happens at once, scale-in waits 1 step after the last change.

🔍 Diagnose

Log type	Platform	Where it goes	Notes
Application logging	Windows, Linux	file system or Blob (Windows)	file-system logging on Windows switches off after 12 h
Web server logging	Windows	file system or Blob	raw HTTP request data (W3C format)
Detailed error messages	Windows	file system	.htm pages for HTTP 400+ errors
Failed request tracing	Windows	file system	IIS trace per failed request (freb)
Deployment logging	both	file system	always on, shown in the Deployment Center
Diagnostics settings	both	Log Analytics, Storage, Event Hubs	AppServiceHTTPLogs, AppServiceConsoleLogs, AppServiceAppLogs tables

⚠️ HTTP 503 or 'Application Error' right after deploying a container

HIGH

COMMON

LIKELY CAUSES

1. Container listens on another port than App Service expects
2. Startup longer than the 230 s default start timeout
3. Image pull failed: identity lacks AcrPull



HOW TO CONFIRM

Read the container start log in the log stream



```
az webapp log tail -n shop-api-prod -g rg-shop
```



FIX

- ✓ Set WEBSITES_PORT to the listening port
- ✓ Raise WEBSITES_CONTAINER_START_TIME_LIMIT (max 1800 s)
- ✓ Grant AcrPull to the app identity and enable acrUseManagedIdentityCreds

⚠️ Swap fails or production slow for a minute after a swap

MEDIUM

OCCASIONAL

LIKELY CAUSES

1. Warm-up path returns an error or redirects (HTTPS-only, auth)
2. Slot settings differ, so staging restarted cold during swap



HOW TO CONFIRM

Activity log of the swap operation; warm-up status codes

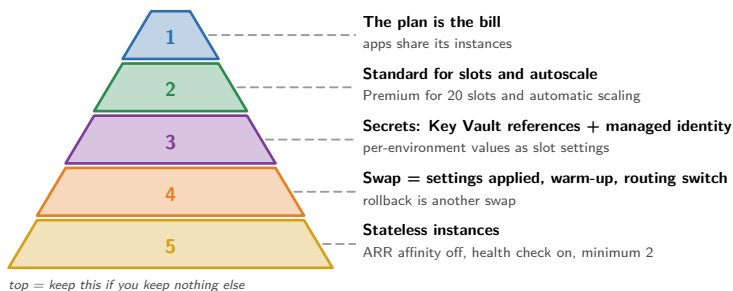


```
az webapp deployment slot swap ... --action preview
```

FIX

- ✓ Set WEBSITE_SWAP_WARMUP_PING_PATH to an anonymous health path
- ✓ Use swap with preview and test before completing

🚩 Recap and self-test



MYTH

- ✗ Each web app in a plan is billed separately

FACT

- ✓ You pay for the plan's instances
Ten small apps can share one S1 plan; isolate a heavy app in its own plan.

- ✗ A swap copies production's custom domain to staging

- ✓ Domains, TLS bindings and scale settings stay with the slot
Only content and non-sticky settings move.

Q1 Cheapest tier that supports deployment slots?

- A Standard (5 slots); Premium v3 and Isolated v2 give 20.

Q2 How do you keep a staging connection string from moving into production on swap?

- A Mark it as a deployment slot setting (--slot-settings in the CLI).

Q3 Which setting tells App Service the port of a custom container?

- A WEBSITES_PORT.

Q4 How does an app setting read a Key Vault secret without code?

- A Value @Microsoft.KeyVault(SecretUri=...) resolved with the app's managed identity (Key Vault Secrets User).

Q5 Validate a swap before it completes?

- A Swap with preview: phase 1 applies target settings and warms up, then complete or cancel.

Q6 Send 10 % of users to the staging slot?

- A Traffic % routing (az webapp traffic-routing set --distribution staging=10), pinned by the x-ms-routing-name cookie.

Q7 Scale up vs scale out?

- A Up = bigger instance size or tier; out = more instances of the same size.

Q8 Environment variable prefix for an Azure SQL connection string?

- A SQLAZURECONNSTR_<name>.

Q9 Why turn ARR affinity off?

- A Stateless apps then spread evenly across instances and survive instance loss without sticky sessions.