CHECKLIST

Azure VM Right-Sizing and Autoscaling Checklist

A practical checklist for validating workload behavior, choosing between right-sizing and autoscaling, and verifying changes before production use.

Azure VM Right-Sizing and Autoscaling Checklist

Use this checklist to decide whether to right-size Azure virtual machines, enable autoscaling, or leave the workload as-is until more evidence is available.

---

1) Confirm the workload is a fit

- [ ] The workload has been running long enough to observe normal and peak behavior.
- [ ] You have a representative measurement window that includes busy periods, quiet periods, and known spikes.
- [ ] You can identify whether the workload is steady, bursty, or schedule-based.
- [ ] The application?s state model is understood:
- [ ] Stateless or mostly stateless
- [ ] Stateful
- [ ] Mixed
- [ ] You know whether the workload can tolerate VM resize events or instance churn.
- [ ] You have a rollback plan for both size changes and scaling-policy changes.

---

2) Collect the right signals

Do not rely on CPU alone.

Performance and capacity signals

- [ ] Average CPU utilization
- [ ] CPU spikes and sustained saturation periods
- [ ] Memory pressure, paging, or swap activity
- [ ] Disk read/write latency
- [ ] Disk queue depth or storage backlog
- [ ] Network throughput and packet behavior
- [ ] Application response time
- [ ] Request queue length or backlog
- [ ] Error rate during peak load
- [ ] Job duration or batch completion time

Operational context

- [ ] Peak periods are tied to business hours, reports, deployments, or batch windows.
- [ ] Short bursts are visible in the data, not hidden by averages.
- [ ] The observed data reflects normal production behavior, not only lab testing.
- [ ] Hidden dependencies have been checked:
- [ ] Database throughput
- [ ] Shared caches
- [ ] Session state
- [ ] Message queues

- [ ] Storage latency

---

3) Decide whether right-sizing is appropriate

Right-size when the VM is consistently larger than the workload needs and the application can safely tolerate a smaller size.

- [ ] CPU is low across a representative window.
- [ ] Memory headroom remains healthy during peak use.
- [ ] Storage latency stays acceptable under load.
- [ ] Network throughput is well below saturation.
- [ ] There is no evidence of bursty demand that would be harmed by reducing headroom.
- [ ] The workload does not depend on temporary overprovisioning to mask another bottleneck.
- [ ] A smaller size still meets latency, throughput, and reliability targets.
- [ ] The resize path is understood:
- [ ] Hot resize supported
- [ ] Restart required
- [ ] Maintenance window needed
- [ ] A rollback to the original size is possible if performance regresses.

Avoid right-sizing if

- [ ] The workload is already close to a storage, memory, or application bottleneck.
- [ ] Usage varies sharply and unpredictably.
- [ ] Metrics are incomplete or unreliable.
- [ ] The VM supports critical tasks that need extra headroom for failover or recovery.

---

4) Decide whether autoscaling is appropriate

Autoscale when demand changes predictably and the workload can add or remove instances safely.

- [ ] The application can run on multiple instances.
- [ ] State is externalized, shared, or replicated safely.
- [ ] Instances can be drained before removal.
- [ ] Long-lived sessions will not be broken unexpectedly.
- [ ] Background jobs can move or retry cleanly.
- [ ] Scale-out signals are well understood.
- [ ] Scale-in behavior is conservative enough to avoid oscillation.
- [ ] The scale policy is based on workload demand, not a single misleading metric.
- [ ] At least one of these is a reliable trigger:
- [ ] Request queue length
- [ ] Concurrent sessions
- [ ] Application latency
- [ ] CPU for CPU-bound workloads
- [ ] Throughput for predictable traffic patterns
- [ ] There is enough time for new instances to become useful before demand peaks.
- [ ] There is enough warm capacity to avoid repeated scaling events.

Avoid autoscaling if

- [ ] The workload is highly stateful and cannot drain cleanly.
- [ ] The real bottleneck is database, storage, or application design.
- [ ] Scaling down would drop active work or active sessions.
- [ ] The environment lacks telemetry to validate scale decisions.

---

5) Choose the right approach

- [ ] Choose right-sizing if the problem is stable overprovisioning.
- [ ] Choose autoscaling if the problem is repeatable demand spikes.
- [ ] Choose both if the baseline is too large and peaks still need more capacity.
- [ ] Choose neither for now if the workload is poorly instrumented, highly stateful, or blocked by another bottleneck.

Quick decision test

- [ ] If average utilization is low but peaks matter, do not shrink based on average alone.
- [ ] If traffic changes predictably, scaling may help more than a larger static VM.
- [ ] If the bottleneck is not CPU, avoid CPU-driven decisions.
- [ ] If the app cannot tolerate instance churn, do not force autoscaling.

---

6) Validate before production

Test one change at a time.

- [ ] Change only one dimension first:
- [ ] VM size
- [ ] Scale policy
- [ ] Both, only after isolated validation
- [ ] Compare the same workload window before and after the change.
- [ ] Measure latency, error rate, and saturation after the change.
- [ ] Verify that throughput remains acceptable under peak load.

- [ ] Confirm memory and storage behavior did not regress.
- [ ] Confirm scaling does not create oscillation or thrashing.
- [ ] Confirm scale-in does not interrupt active work.
- [ ] Confirm rollback works in a real test.
- [ ] Record the final baseline for future comparison.

---

7) Production readiness checklist

- [ ] Monitoring is in place for CPU, memory, storage, network, and application latency.
- [ ] Alerts exist for saturation and abnormal error rates.
- [ ] A maintenance window is scheduled if the resize requires downtime.
- [ ] Recovery objectives still make sense after the change.
- [ ] Backup and restore behavior have been considered.
- [ ] Failover, if relevant, still has enough headroom.
- [ ] The team knows who approves scaling policy changes.
- [ ] The team knows who can revert a resize or scale rule.
- [ ] Documentation is updated with the new baseline and operating assumptions.

---

8) Final go/no-go decision

Go if all are true

- [ ] The workload has enough telemetry.
- [ ] The bottleneck is understood.
- [ ] The chosen approach matches the workload pattern.

- [ ] The application can tolerate the change.
- [ ] Validation showed no meaningful regression.
- [ ] Rollback is ready.

No-go if any are true

- [ ] The evidence is incomplete.
- [ ] The workload is too stateful for the planned autoscaling model.
- [ ] A different bottleneck is the real problem.
- [ ] Performance regressed during testing.
- [ ] There is no safe rollback path.

---

9) Operational notes to keep on file

- [ ] Current VM size or fleet baseline: _____
- [ ] Primary bottleneck: _____
- [ ] Representative measurement window: _____
- [ ] Peak demand pattern: _____
- [ ] Scale-out trigger: _____
- [ ] Scale-in guardrail: _____
- [ ] Rollback owner: _____
- [ ] Review date: _____

Summary

Right-sizing reduces static waste. Autoscaling absorbs repeatable demand spikes. Both depend on real workload behavior, not averages alone. If the workload is not well understood, pause and gather better evidence before changing production capacity.