

- 1 MCP DevOps Governance Rollout Kit (Platform Teams)
 - 1.1 Executive Summary
 - 1.2 What Changed (Why This Playbook Exists Now)
 - 1.3 Critical Accuracy Note (Use This Language Internally)
 - 1.4 Non-Negotiables
 - 1.5 Step 1: Build a Server Inventory Before You Roll Out Anything
 - 1.5.1 MCP Server Inventory Worksheet (Template)
 - 1.5.2 Risk Tiers (Recommended)
 - 1.6 Step 2: Decide Transport Policy by Action Tier
 - 1.6.1 Transport Policy Matrix (Example)
 - 1.6.2 Practical Call
 - 1.7 Step 3: Standardize Client Config Patterns (Conceptual)
 - 1.7.1 Local Pilot Config (Read-Only First)
 - 1.7.2 Remote Shared Service Config (Conceptual)
 - 1.8 Step 4: Design Identity and Authorization Before Tool Coverage Expands
 - 1.8.1 Identity Principles
 - 1.8.2 Recommended Model
 - 1.9 Step 5: Implement Action Governance and Approval Gates
 - 1.9.1 Approval Policy Pattern (Conceptual)
 - 1.9.2 Operational Rule
 - 1.10 Step 6: Log MCP Actions Like a New API Surface (Because They Are)
 - 1.10.1 Minimum Log Fields
 - 1.10.2 SIEM Event Schema (Conceptual)
 - 1.10.3 Alerting Ideas
 - 1.11 Step 7: Pilot Use Cases in the Right Order
 - 1.11.1 Tier 1 (Read-Only) - Start Here
 - 1.11.2 Tier 2 (Reversible Writes)
 - 1.11.3 Tier 3 (Infra Mutation with Human Gate)
 - 1.11.4 Tier 4 (Destructive / High Blast Radius)
 - 1.12 Step 8: Roll Out in Phases (Suggested 6-8 Week Plan)
 - 1.12.1 Phase 0 (Week 0-1) - Inventory and Policy Baseline
 - 1.12.2 Phase 1 (Week 1-2) - Read-Only Pilot
 - 1.12.3 Phase 2 (Week 3-4) - Reversible Writes with Approval Conditions
 - 1.12.4 Phase 3 (Week 5-6) - Controlled Infra Mutations
 - 1.12.5 Phase 4 (Week 7+) - Expansion with Continuous Review
 - 1.13 Step 9: Incident Runbook (Wrong Target / Unsafe Action)
 - 1.14 Step 10: Use a Due-Diligence Checklist for Every MCP Server
 - 1.15 Weekly Governance Review Template
 - 1.16 Recommended Messaging for Leadership / Governance Boards
 - 1.17 Sources (Verified February 25, 2026)
 - 1.18 Final Recommendation

1 MCP DevOps Governance Rollout Kit (Platform Teams)

Updated: February 26, 2026

Author: Mathieu Kessler

Companion article: <https://www.talk-nerdy-to-me.com/blog/mcp-usb-c-of-devops-governance-playbook>

1.1 Executive Summary

MCP is moving from demo protocol to operational integration layer for cloud and DevOps tooling. The limiting factor is no longer “can an agent call the tool?” The limiting factor is whether your platform team has a safe operating model for:

- server trust boundaries
- transport choices (`stdio` vs `streamable-http`)
- identity and authorization scopes
- approval gates for risky actions
- auditability / SIEM logging
- rollout sequencing and kill switches

This playbook gives you a practical rollout framework for introducing MCP servers without turning infrastructure operations into an uncontrolled experiment.

1.2 What Changed (Why This Playbook Exists Now)

The Q1 2026 inflection point is a productization shift, not just a protocol shift:

- Azure MCP Server reached stable 1.0
- Azure Functions MCP support matured, with an important distinction between GA extension support and preview self-hosted MCP server hosting
- Azure DevOps local MCP server reached GA
- Visual Studio 2026 includes Azure MCP tooling and agentic cloud workflows
- Terraform MCP is documented by HashiCorp (beta) with explicit security guidance

This means platform teams now need governance patterns, not just curiosity.

1.3 Critical Accuracy Note (Use This Language Internally)

Do **not** say “Azure Functions MCP is GA” without context.

Use this instead:

- **Azure Functions OpenAI extension MCP support is GA** (Microsoft Learn / Azure Functions docs)
- **Hosting SDK-based self-hosted MCP servers on Azure Functions is public preview** (Microsoft Learn and Apps on Azure blog)

That distinction matters for change management, support expectations, and production risk reviews.

1.4 Non-Negotiables

- Tier 3 and Tier 4 actions require human approval and a documented execution authority (CI/control plane or break-glass path).
- Every MCP server has an owner, version pin, and rollback / kill-switch procedure.
- Tool scope is intentionally limited for pilot phases; no “enable everything” defaults.
- Authentication and authorization are defined before expanding tool coverage.
- MCP actions are logged with actor, tool, target, credential context, approval status, and result.
- Prefer short-lived credentials and managed identities over static secrets in local config.

1.5 Step 1: Build a Server Inventory Before You Roll Out Anything

Start with an inventory, not prompts.

1.5.1 MCP Server Inventory Worksheet (Template)

Server	Owner	Hosting	Transport	AuthN/AuthZ	Tool Scope	Risk Tier	Prod Enabled	Logging	Version Pin	Notes
terraform-mcp	Platform IaC	local (pilot)	stdio	local cloud creds + role	plan/apply/workspace	Tier 2/3	No	local -> SIEM	0.x pinned	apply blocked in prod
azure-mcp-server	Platform Cloud	internal VM/App	streamable-http	Entra app + RBAC	inventory/deploy/read ops	Tier 1/2	Non-prod	central logs	1.x pinned	per-subscription scopes
azure-devops-mcp	DevEx	local	stdio	PAT / Entra (scoped)	Boards/Repos/Pipelines/Wiki	Tier 1/2	N/A	local -> SIEM	pinned	tool domains scoped
billing-mcp	FinOps	internal service	streamable-http	IAM role + read-only	cost/budgets/anomalies	Tier 1	Yes	central logs	pinned	read-only only

1.5.2 Risk Tiers (Recommended)

- **Tier 1:** Read-only / no state mutation
- **Tier 2:** Reversible writes (tickets, comments, PR drafts, non-prod reruns)
- **Tier 3:** Infrastructure mutations / deploys / restarts / plan+apply workflows
- **Tier 4:** Destructive or high-blast-radius operations (delete, destroy, prod failover, data-destructive actions)

1.6 Step 2: Decide Transport Policy by Action Tier

Do not treat transport as a preference toggle. Treat it as policy.

1.6.1 Transport Policy Matrix (Example)

Action Tier	Default Transport	Allowed Remote?	Extra Controls Required
Tier 1 (read-only)	stdio or streamable-http	Yes	auth, logging, rate limits
Tier 2 (reversible write)	stdio preferred	Yes, after review	authz scopes, owner routing, audit logs
Tier 3 (infra mutation)	stdio for pilot; CI executor for prod	Limited	human approval, change record, least privilege, rollback path
Tier 4 (destructive/prod critical)	no direct agent execution by default	Exceptional only	break-glass workflow, multi-party approval, recorded session

1.6.2 Practical Call

- Use **stdio** when minimizing network exposure and keeping credentials local matters most.
- Use **streamable HTTP** when shared services, central policy, and auditability matter most.
- Never rely on transport choice as your only control.

1.7 Step 3: Standardize Client Config Patterns (Conceptual)

1.7.1 Local Pilot Config (Read-Only First)

```
{
  "mcpServers": {
    "terraform": {
      "command": "terraform-mcp-server",
      "args": ["serve"],
      "env": {
        "TF_MCP_READ_ONLY": "true",
        "TF_MCP_LOG_LEVEL": "info"
      }
    },
    "azure-devops": {
      "command": "azdo-mcp",
      "args": ["serve"],
      "env": {
        "AZDO_ORG_URL": "https://dev.azure.com/your-org",
        "AZDO_TOOL_DOMAINS": "repos,boards,pipelines"
      }
    }
  }
}
```

Guidance:

- Default to read-only on first enablement.
- Scope tool domains / capabilities per server.
- Export logs locally and forward to your SIEM/collector.
- Pin versions; don't let each workstation drift independently.

1.7.2 Remote Shared Service Config (Conceptual)

```
{
  "mcpServers": {
    "platform-azure": {
      "transport": {
        "type": "streamable-http",
        "url": "https://mcp.platform.example.com/azure/mcp"
      },
      "headers": {
        "Authorization": "Bearer <short-lived-token>"
      }
    },
    "billing": {
      "transport": {
        "type": "streamable-http",
        "url": "https://mcp.platform.example.com/billing/mcp"
      }
    }
  }
}
```

Guidance:

- Use enterprise auth (Entra / IAM / SSO-integrated tokens).
- Prefer short-lived tokens over static secrets.
- Enforce authorization on the server side, not just client config.
- Log every invocation centrally.

1.8 Step 4: Design Identity and Authorization Before Tool Coverage Expands

1.8.1 Identity Principles

- **AuthN first:** Decide how the human+agent session authenticates before enabling write tools.
- **AuthZ by domain:** Scope to subscriptions/projects/namespaces/repos/pipelines rather than broad admin roles.
- **Short-lived credentials:** Prefer federated or managed identity patterns over stored secrets.
- **Separate execution authority:** The server can propose or prepare changes; CI/control planes should execute prod mutations.

1.8.2 Recommended Model

- Tier 1 / Tier 2 local pilots can use stdio + scoped local credentials + logging export.
- Tier 3 production mutations should execute through CI or a platform control plane with protected approvals.
- Tier 4 actions remain denied by default except break-glass runbooks.

1.9 Step 5: Implement Action Governance and Approval Gates

This is the part most teams skip and later regret.

1.9.1 Approval Policy Pattern (Conceptual)

```
policies:  
  - name: tier1-read-only  
    match:  
      actionTier: [1]  
    effect: allow  
    requirements:  
      - log_to_siem  
  
  - name: tier2-reversible-writes  
    match:  
      actionTier: [2]  
    effect: allow_with_conditions  
    requirements:  
      - log_to_siem  
      - owner_scope_check  
      - change_ticket_or_issue_link  
  
  - name: tier3-infra-mutation  
    match:  
      actionTier: [3]  
    effect: require_human_approval  
    requirements:  
      - approved_change_request  
      - executor_is_ci_or_control_plane  
      - plan_or_dry_run_artifact  
      - rollback_path_documented  
      - log_to_siem  
  
  - name: tier4-destructive  
    match:  
      actionTier: [4]  
    effect: deny_by_default  
    exceptions:  
      - break_glass_runbook
```

- multi_party_approval
- recorded_session

1.9.2 Operational Rule

If your policy is “engineers should be careful,” you do not have a policy.

1.10 Step 6: Log MCP Actions Like a New API Surface (Because They Are)

Treat MCP traffic as a new control-plane surface.

1.10.1 Minimum Log Fields

- timestamp
- human identity + agent/client identity
- server name/version and transport
- tool name + target system + risk tier
- credential/principal context (no secret values)
- approval requirement and outcome
- result, duration, artifacts, correlation IDs

1.10.2 SIEM Event Schema (Conceptual)

```
{
  "timestamp": "2026-02-25T14:21:30Z",
  "eventType": "mcp.tool_invocation",
  "traceId": "01HS...",
  "sessionId": "agent-session-abc123",
  "actor": {
    "type": "human+agent",
    "humanId": "jane.doe@example.com",
    "agentClient": "ide-agent",
    "agentModel": "vendor/model"
  },
  "server": {
    "name": "terraform-mcp",
    "version": "0.9.2",
    "host": "laptop-123",
    "transport": "stdio"
  },
  "tool": {
    "name": "terraform.plan",
    "riskTier": 3,
    "target": "live/staging/services/payments-api"
  },
  "auth": {
    "principal": "arn:aws:iam::123456789012:role/staging-terraform-plan",
  }
}
```

```
    "authType": "federated-short-lived"
  },
  "approval": {
    "required": true,
    "status": "approved",
    "approver": "platform-oncall@example.com",
    "changeRef": "CHG-4821"
  },
  "result": {
    "status": "success",
    "durationMs": 8430,
    "artifactRef": "s3://.../tfplan-4821"
  }
}
```

1.10.3 Alerting Ideas

- Tier 3/4 action attempted without approval metadata
- server version outside approved list
- repeated failed calls against protected targets
- sudden spike in broad inventory enumeration
- local-only servers invoking production-scope actions

1.11 Step 7: Pilot Use Cases in the Right Order

1.11.1 Tier 1 (Read-Only) - Start Here

Examples:

- cloud inventory lookups
- cost and budget checks
- pipeline status and failure summaries
- cluster health summaries
- documentation and runbook retrieval

1.11.2 Tier 2 (Reversible Writes)

Examples:

- create/update incident tickets
- draft pull requests or change proposals
- annotate alerts with context
- rerun non-prod pipelines
- create remediation checklists

1.11.3 Tier 3 (Infra Mutation with Human Gate)

Examples:

- Terraform plan + CI-mediated apply
- staging deploys via approved pipelines
- Kubernetes rollout restart in non-prod
- controlled scale actions in non-prod

Requirements:

- human approval
- explicit target confirmation
- plan/dry-run artifact
- CI/control-plane execution authority
- rollback path documented

1.11.4 Tier 4 (Destructive / High Blast Radius)

Examples:

- terraform destroy
- deleting production resources
- destructive DB actions
- failover operations

Default policy:

- deny by default
- break-glass only
- multi-party approval
- recorded steps and post-incident review

1.12 Step 8: Roll Out in Phases (Suggested 6-8 Week Plan)

1.12.1 Phase 0 (Week 0-1) - Inventory and Policy Baseline

- inventory MCP servers and tool scopes
- assign owners
- define risk tiers and transport policy
- define logging schema and retention
- define CI/control-plane-only operations

1.12.2 Phase 1 (Week 1-2) - Read-Only Pilot

- enable Tier 1 workflows
- validate audit completeness and auth behavior
- train operators on target verification and prompt boundaries
- measure usefulness and latency

1.12.3 Phase 2 (Week 3-4) - Reversible Writes with Approval Conditions

- enable Tier 2 workflows for select teams

- require owner scope + issue/change references
- run tabletop exercises for wrong-target scenarios

1.12.4 Phase 3 (Week 5-6) - Controlled Infra Mutations

- allow Tier 3 changes with human approval
- force prod execution through CI/control planes
- require plan/dry-run artifacts and rollback paths
- monitor rollback rate and near-misses closely

1.12.5 Phase 4 (Week 7+) - Expansion with Continuous Review

- review server/tool scopes monthly
- remove low-value/high-risk scopes
- pin and test upgrades on schedule
- treat MCP governance as ongoing platform engineering work

1.13 Step 9: Incident Runbook (Wrong Target / Unsafe Action)

1. Contain

- Disable the affected MCP server or revoke its credentials immediately
- Freeze related CI/CD applies/deployments for the impacted domain
- Announce incident channel and single incident commander

2. Preserve evidence

- Collect MCP request/response logs, tool invocation logs, and approvals
- Capture server version, config, and tool scope at incident time
- Preserve cloud/platform audit logs and CI run artifacts

3. Assess impact

- What changed? (resources, configs, tickets, pipelines, secrets references)
- Which environment was affected? (dev/staging/prod)
- Is this reversible with standard workflow, or do you need break-glass?

4. Recover safely

- Prefer forward-fix via reviewed change in CI/control plane
- Use provider-native emergency actions only if incident severity requires it
- Document every manual step and reconcile back to IaC/state after stabilization

5. Prevent recurrence

- Tighten tool scope, credentials, transport policy, or approval requirements

- Add wrong-target checks (env confirmation, resource tags, prod hard-stop)
- Update runbooks and operator training

1.14 Step 10: Use a Due-Diligence Checklist for Every MCP Server

Identity & access

- Does the server support strong authentication for your environment?
- Can you scope authorization by tool/domain/resource?
- Can credentials be short-lived and rotated automatically?

Transport & networking

- Which transports are supported (stdio, streamable HTTP)?
- Is there hardening guidance for remote hosting?
- Can you restrict inbound access (private network, firewall, mTLS, gateway)?

Safety & operations

- Are tool actions clearly documented, including destructive operations?
- Is there dry-run/plan support for risky actions?
- Are logs structured and exportable?
- Are versions pin-able with release notes/changelogs?
- Is failure behavior documented (timeouts/retries/idempotency)?

Governance

- Can you limit exposed tools to a subset for pilot?
- Can you map actions to risk tiers and approval policy?
- Do you have a rollback/kill-switch procedure for this server?

1.15 Weekly Governance Review Template

Measure safety and value together.

Adoption

- Active users this week:
- Active MCP servers:
- Top 5 workflows by volume:

Safety

- % of tool invocations logged with complete fields:
- Tier 3/4 actions without approval (target = 0):
- Wrong-target or near-miss events:
- Rollbacks triggered:

Quality

- Useful outcome rate (agent action reduced manual work):
- False positive / hallucinated action proposals:

- Mean time to approved change (with MCP vs baseline):

Cost / Operations

- MCP server uptime and error rate:
- Token/model spend (if applicable):
- Support tickets caused by MCP tooling:

Decisions

- Promote to next phase? (yes/no)
- Which server/tool scopes expand next week?
- Which scopes get frozen or rolled back?

1.16 Recommended Messaging for Leadership / Governance Boards

Use this wording:

MCP is not a permission to let agents run production operations. It is a protocol for tool interoperability. We are rolling it out under risk-tiered policy, approval gates, and audit logging just like any other control-plane capability.

This frames the program correctly and reduces pressure to over-automate before guardrails exist.

1.17 Sources (Verified February 25, 2026)

Primary and vendor sources:

- Anthropic: Introducing the Model Context Protocol
<https://www.anthropic.com/news/model-context-protocol>
- MCP Docs: Architecture / transports
<https://modelcontextprotocol.io/docs/learn/architecture>
- Azure SDK Blog: Azure MCP Server GA (stable 1.0)
<https://devblogs.microsoft.com/azure-sdk/azure-mcp-server-is-now-generally-available/>
- Microsoft Learn: Azure MCP Server in Visual Studio 2026
<https://learn.microsoft.com/en-us/visualstudio/azure/mcp-server>
- Visual Studio Blog: agentic cloud workflow + CI/CD generation
<https://devblogs.microsoft.com/visualstudio/building-cloud-apps-just-got-easier-just-got-agentic/>
- Azure Functions MCP docs (extension GA + self-hosted preview)
<https://learn.microsoft.com/en-us/azure/azure-functions/functions-model-context-protocol>
- Azure Functions self-hosted MCP tutorial (.NET, preview)
<https://learn.microsoft.com/en-us/azure/azure-functions/functions-develop-mcp-server>
- Apps on Azure Blog: remote MCP servers on Azure Functions (public preview)
<https://techcommunity.microsoft.com/blog/appsonazureblog/building-remote-mcp-servers-with-azure-functions-public-preview/4400088>

- Apps on Azure Blog: Azure Functions OpenAI extension + MCP support (Ignite updates)
<https://techcommunity.microsoft.com/blog/appsonazureblog/introducing-azure-functions-openai-extension-model-context-protocol-support-for-buil/4304473>
- Azure DevOps Blog: local MCP server GA
<https://devblogs.microsoft.com/devops/azure-devops-local-mcp-server-is-now-generally-available/>
- HashiCorp: Terraform MCP Server docs (beta)
<https://developer.hashicorp.com/terraform/tools/mcp-server>
- HashiCorp: Terraform MCP security configuration
<https://developer.hashicorp.com/terraform/tools/mcp-server/configure-security>
- AKS Blog: AKS MCP Server and AKS Agent
<https://blog.aks.azure.com/2025/08/20/aks-managed-mcp-server-and-aks-agent>
- AWS Docs: Billing & Cost Management MCP Server
<https://docs.aws.amazon.com/cost-management/latest/userguide/what-is-bcm-mcp-server.html>

Ecosystem snapshot (used for market context, not authoritative maturity certification):

- StackGen: Top 10 MCP Servers for Platform Engineers
<https://stackgen.com/blog/top-10-mcp-servers-for-platform-engineers>

1.18 Final Recommendation

Treat MCP enablement like introducing a new platform control-plane surface.

If you do that, you can capture the speed benefits without losing operational control. If you don't, you're just moving infrastructure risk behind a chat prompt.