CLOUDBOLT

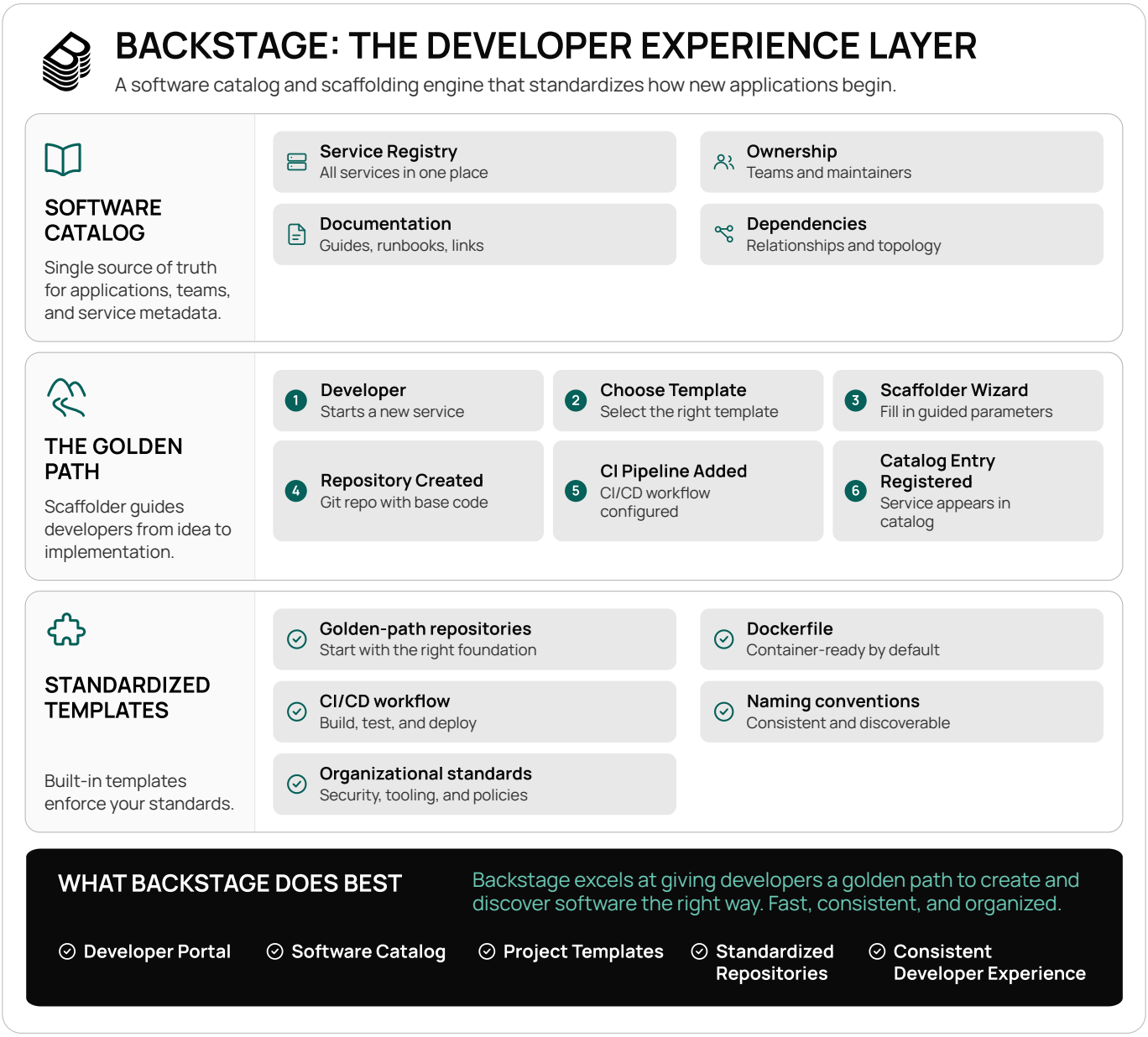
COMPLEMENTARY STACK SERIES: PART 4

Backstage and CMP: A Catalog Is Not a Platform

Internal developer platforms raise a different version of the same question. A team that already runs Backstage has a software catalog, a scaffolder that spins up new services from templates, and a permissions framework governing who can use them. That looks a lot like a self-service front end, so it is fair to ask why a cloud management platform belongs in the same stack. The honest answer starts with what Backstage was actually built to solve, and it is a smaller problem than the one a governance layer has to cover.

WHAT BACKSTAGE ACTUALLY DOES WELL

Backstage's core value is the software catalog: a single, searchable source of truth for what services exist, who owns them, and how they relate to each other. The Scaffolder is the engine on top of that catalog, and it earns its keep. Software templates are pre-defined skeletons a developer can fill in through a guided wizard, and the Scaffolder executes them: creating a new repo with a baseline Dockerfile, a CI workflow, and the matching catalog entry, all in the org's standard shape. A permissions framework controls which parameters and steps in a template a given user can touch. For the specific job of giving developers a golden path into a new service instead of copying a five-year-old repo, that combination genuinely works.



WHAT THE CATALOG AND SCAFFOLDER ADD…AND WHAT THEY DON'T

Backstage assumes the execution layer already exists beneath it. It does not ship its own engine for provisioning infrastructure, enforcing budget, or running approvals; every action past creating a repo and a catalog entry has to be wired to a plugin that calls something else, and each of those integrations is custom, maintained in-house, and prone to becoming what one platform engineering analysis calls a fragile messy middle of direct wiring into CI/CD, GitOps, and Kubernetes. Industry commentary on the project has been blunt about the boundary: Backstage solved the portal problem, not the platform problem, and strictly speaking it is not itself an internal developer platform because it lacks operational features beyond documentation and base templating. That is not a criticism of the tool. It is a description of what a catalog and a scaffolder are for.

WHERE CMP PICKS UP

A cloud management platform is built for exactly the layer Backstage assumes is already there. CloudBolt's catalog handles cross-cloud orchestration, automated approval chains, and cost estimates enforced before anything deploys, connecting to Terraform, Ansible, cloud APIs, and 200-plus other systems as the actual execution behind a request. Third-party analysis of this category split describes it the same way from the other direction: cloud management platforms are ops-first, built around governance, compliance, and cost control, while internal developer platforms are dev-first, built around removing friction for the people shipping code. Those are different jobs with different owners, not competing answers to the same one.

THE PRACTICAL PAIRING

In practice, a Backstage template's 'create environment' button does not have to be backed by a hand-built script talking directly to a cloud API. It can call CMP instead, which runs the request through the same approval, tagging, and cost logic governing every other resource in the estate, then hands back a working environment. Backstage still owns the catalog entry, the documentation, and the developer-facing wizard. CloudBolt still owns what actually happens when that wizard submits, and the governed record of the result afterward. Replacing a fragile custom plugin with a governed API call does not cost a platform team anything it was relying on.

BACKSTAGE + CLOUDBOLT: BETTER TOGETHER

Backstage creates the developer experience. CloudBolt governs everything that happens after the button is clicked.

1 BACKSTAGE

The Developer Experience Layer

- Software Catalog
- Templates
- Documentation
- Ownership & Teams

Create New Environment — Developers discover, choose a template, and request a new environment.

2 CLOUDBOLT CMP

The Governance & Orchestration Layer

- Service Catalog
- Approval Workflow
- Cost Estimate
- Governance & Policy
- Policy Engine
- Blueprints
- Multi-tool Orchestration
- API Integrations

CloudBolt governs the request, enforces policy, estimates cost, and orchestrates the work.

3 EXECUTION LAYER

Tools & Systems Orchestrated by CloudBolt

- Terraform — Provision Infrastructure
- Ansible — Configure Systems
- Cloud APIs — Public & Private Clouds
- ServiceNow — ITSM / Ticketing
- GitHub — Source Code & Repos
- Kubernetes — Deploy & Orchestrate

CloudBolt orchestrates whichever tools are required to fulfill the request.

4 DELIVERED ENVIRONMENT

Ready. Governed. Discoverable.

- Repository
- Infrastructure
- Configuration
- Documentation
- Ownership
- Catalog Entry

Ready · Governed · Discoverable — Developers get a fully provisioned, configured, and discoverable environment.

WHO DOES WHAT

CLOUDBOLT CMP	BACKSTAGE
Service Catalog	Developer Portal
Approvals & Workflows	Software Catalog
Governance & Policy	Templates
Cost Controls & Estimates	Scaffolder (Create Workflows)
Multi-tool Orchestration	Documentation
Policy Enforcement	Ownership & Teams
Lifecycle Management (Day Two)	Developer Experience
Cross-Cloud & Tool Integrations	Golden Path & Standards

CloudBolt doesn't replace Backstage.

Backstage remains the developer portal. CloudBolt replaces the fragile custom glue between Backstage and the infrastructure.

Together they deliver a better developer experience and enterprise governance—without compromising either.

Better Developer Experience | Governed Requests | Standardized Provisioning | Cross-Tool Automation | Enterprise Governance

THE BOTTOM LINE

Backstage's catalog and scaffolder make it easier to standardize how developers create and discover services, which is the job a developer portal exists to do.

They do not add provisioning, budget enforcement, or cross-tool governance underneath that portal, because that was never the layer Backstage was designed to own. This series closes next with the synthesis version of that argument: how to describe the full stack, Ansible, Terraform, Backstage, and CMP together, in one conversation instead of four separate ones.

Talk to CloudBolt cloudbolt.io/demo