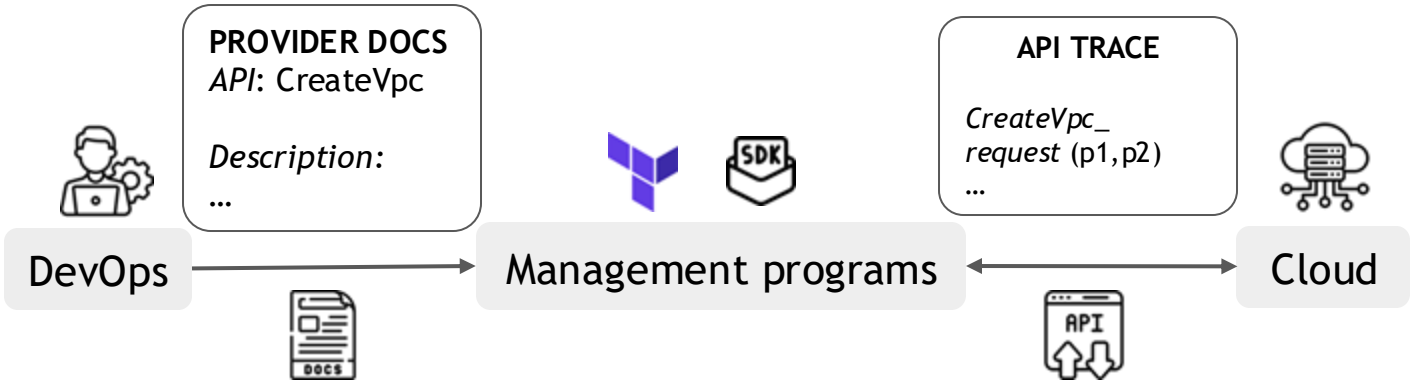
A Case for Learned Cloud Emulators

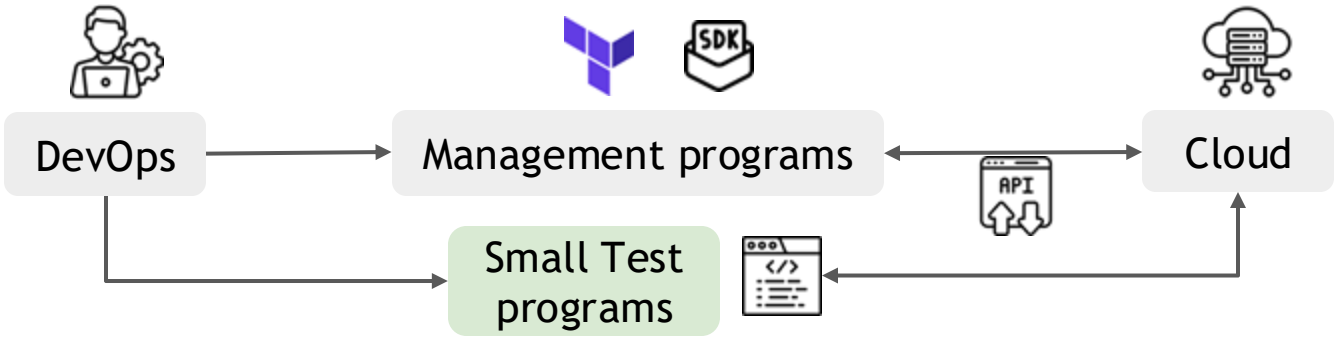
Archit Bhatnagar, Yiming Qiu, Sarah McClure, Sylvia Ratnasamy, Ang Chen

Cloud infra is managed programmatically



DevOps use **management programs** to provision/maintain resources at scale

But, testing Management Programs is hard

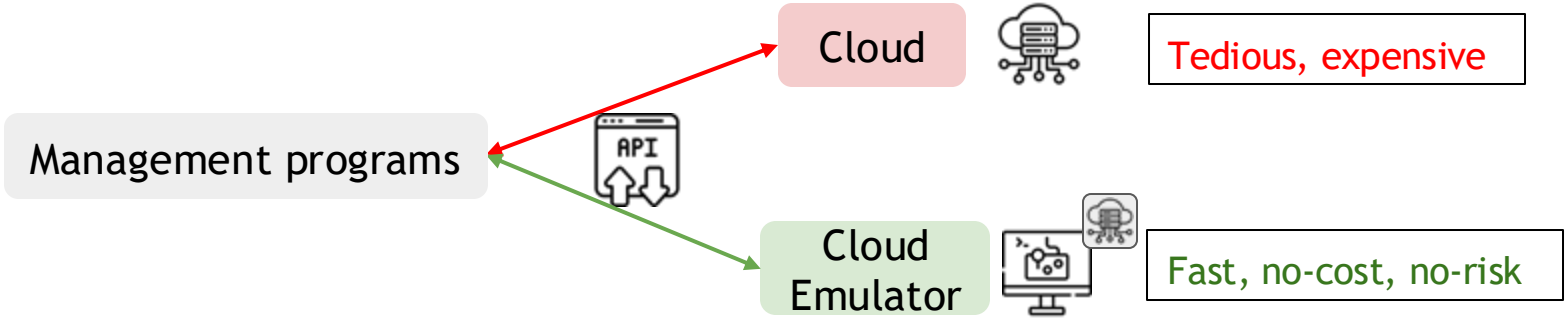


DevOps debug/test mgmt. programs by **running them against the cloud**

Necessary, but tedious (long prov. times) and expensive[1]
Testing directly on cloud impacts development **cost**, **scale** and **velocity**

1. How can I experiment with Cloud without going broke?, *DevOps Stack Exchange*

Cloud emulators are gaining popularity



Emulation: Mock the API request/response behavior
→ Similar to Mininet for setting up virtual networks

SOTA: Emulator for AWS LocalStack ☆ Star 63k

"Emulate cloud services locally. Build fast. Test safely. No surprises."

Problem: Today's emulators are built manually

Labor-intensive

➔ Low API coverage

Not future proof

➔ New services/APIs get added frequently, hard to maintain

Error-prone

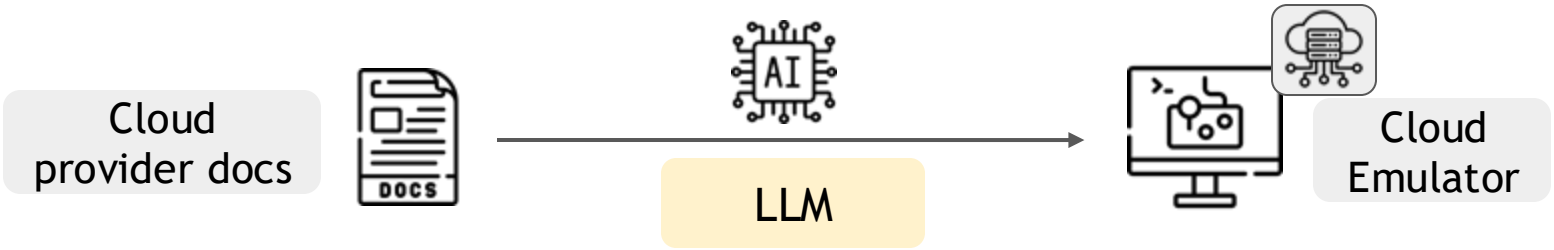
➔ Can have behavior inconsistent with cloud[2]

2. Fidelity of Cloud Emulators: The Imitation Game of Testing Cloud-Based Software, *ICSE'25*

Can LLMs automate emulator generation?

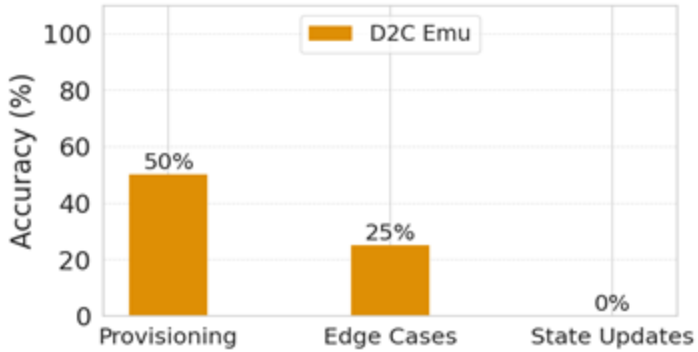
Naive Direct to code(D2C) :

→ use LLM to generate emulation code directly



D2C emulator suffers from inaccuracies

→ missing resources, checks, etc.



Path to Better Automated Emulation

Goal is building a tool that :

→ is automated, can correctly emulate the cloud APIs (with high coverage)

→ keeps updating/refining itself based on the present cloud behavior

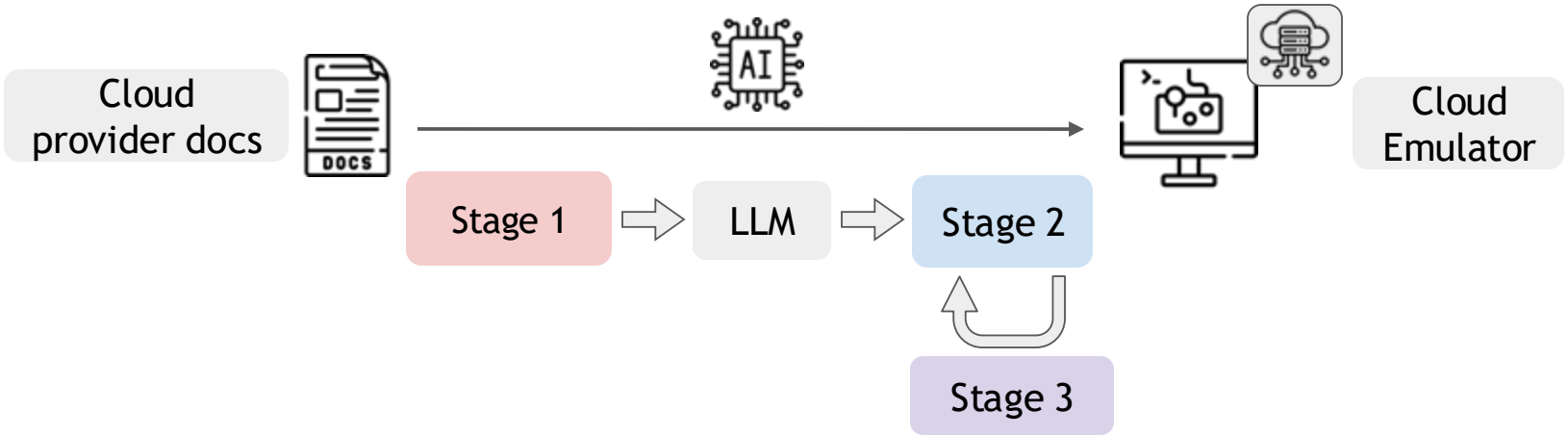
Challenges -

1) Learning the right information given large documents

2) Converting the information into high-fidelity emulation code

3) Adapting with changing cloud services and generation errors

Building the Emulator Generation Workflow



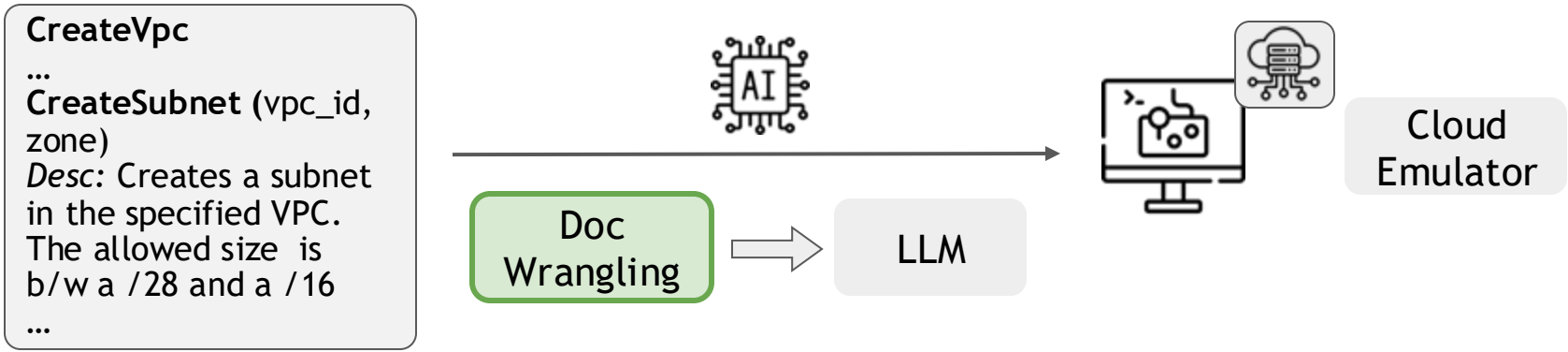
1) Learning the right information given large documents

2) Converting the information into high-fidelity emulation code

3) Adapting with changing cloud services and generation errors

Stage 1: Doc Wrangling for extracting info

Challenge: Large amount of cloud info and limited context window for LLMs



Cloud provider API docs are structured by resources

Docs list API descriptions, params — detailed as users rely on them

Limit Context with Resource-Aware Pruning

CreateVpc
…….

CreateSubnet (vpc_id, zone)
Desc: Creates a subnet in the specified VPC.
…
DeleteSubnet (subnet_id)
…

Per-Resource Info

Dependency Graph based pruning

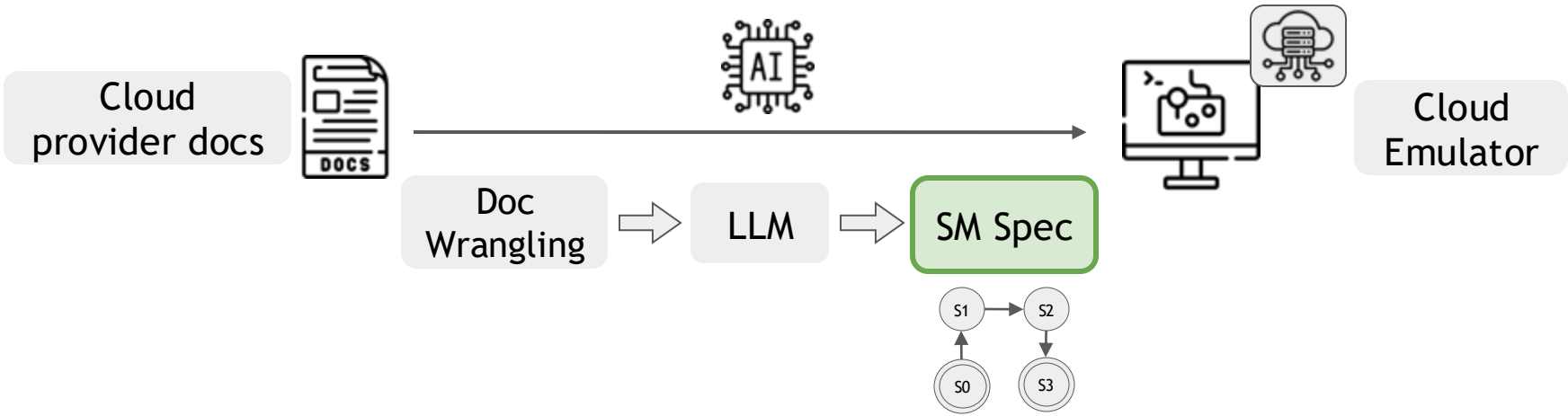
Key insights

Symbolic parsing
—semi-structured docs

Context pruning
—resource-centric info, identify correlation

Stage 2: State Machine Specification

Challenge: Generating high fidelity structured code



Cloud — hierarchical state-machine, API call = transition

State Machine (SM) abstraction helps structured generation

SM Spec Constraints Generated Emulator Code

Subnet.txt:

CreateSubnet (vpc_id, zone)
Desc: Creates a subnet in the specified VPC.

The allowed size is b/w a /28 and a /16 netmask

…

Dependency Graph

Grammar

LLM

```
SM Vpc { States: cidr: str
 … }

SM Subnet {
  States: vpc_id: str, cidr: str
  Transitions:
     CreateSubnet(vpc_id, cidr) {
        assert(16 <= mask(cidr) <= 28);
     }
…
```

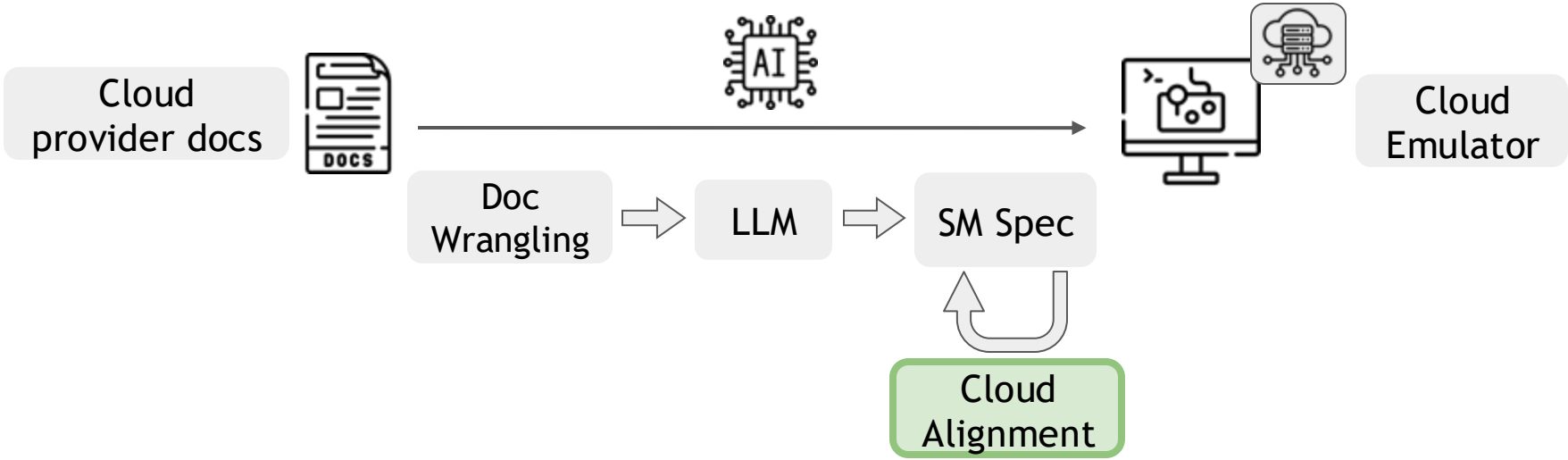
Key insights

Representation to constraint:
— Grammar spec, map intent to action

Incremental Generation
— iterate, generate, leave stubs

Stage 3: Automated Alignment

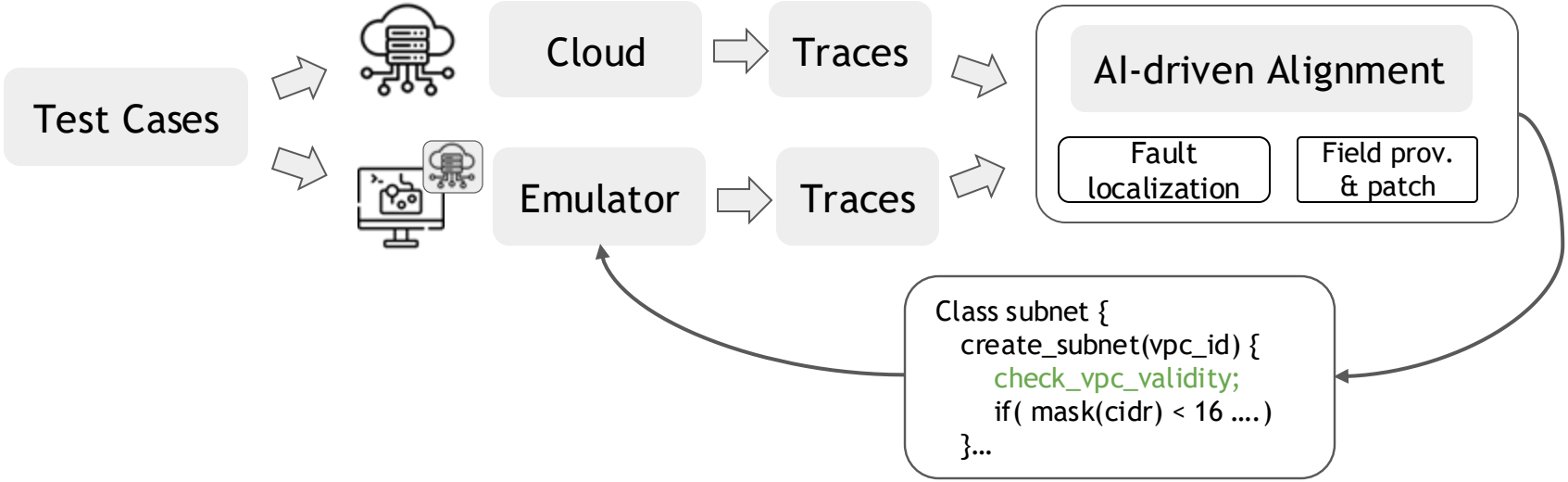
Challenge: Gaps b/w generated emulator and cloud, precise fault isolation



Cloud API traces are verbose to enable **alignment engine**

Can fix test errors, isolating issues and patching logic for **emulator**

Find and Patch Inconsistencies Automatically



Key insights

Fault Localization
— Trace API logs → graph similarity to identify differing log

State provenance for patching
— writers APIs for state, maintain logs

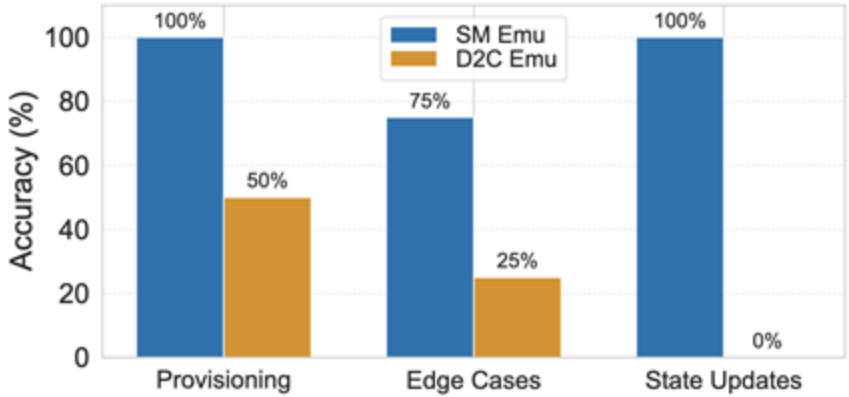
Preliminary Evaluations

Compared CloudEmu (w/o alignment) against 2 baselines–

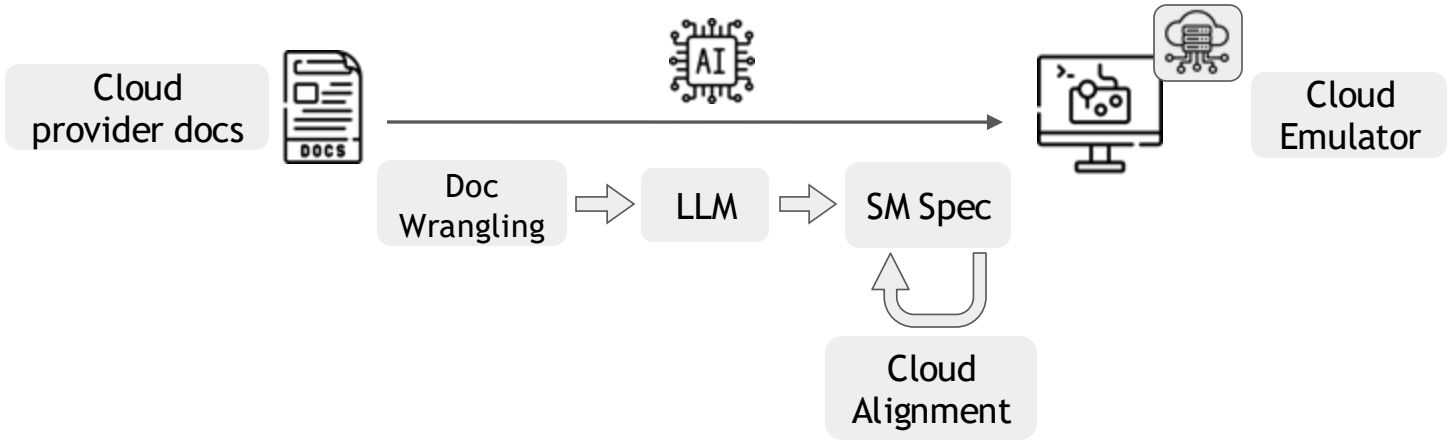
Manual engineering *(Localstack)*:
— CloudEmu outperforms on coverage

Service	CloudEmu API Coverage	LocalStack API Coverage
Network Firewall	100%	11%

D2C emulator:
Issues with state (missing attributes) and transitions (missing/invalid checks)



Summary and Takeaways



Development of modern cloud infrastructure needs emulation for better testing

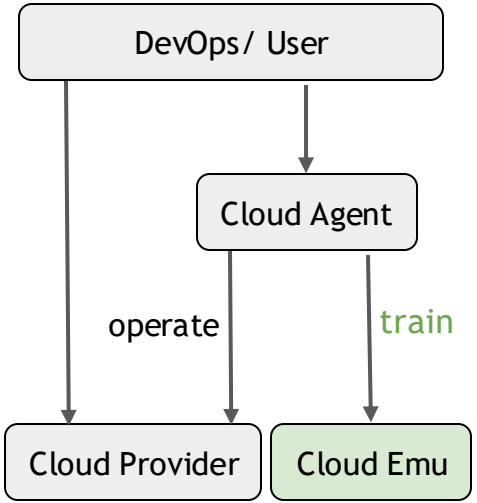
Case for using LLMs to enable emulator code generation

Modeling cloud resources as state machines ⇒ principled code generation

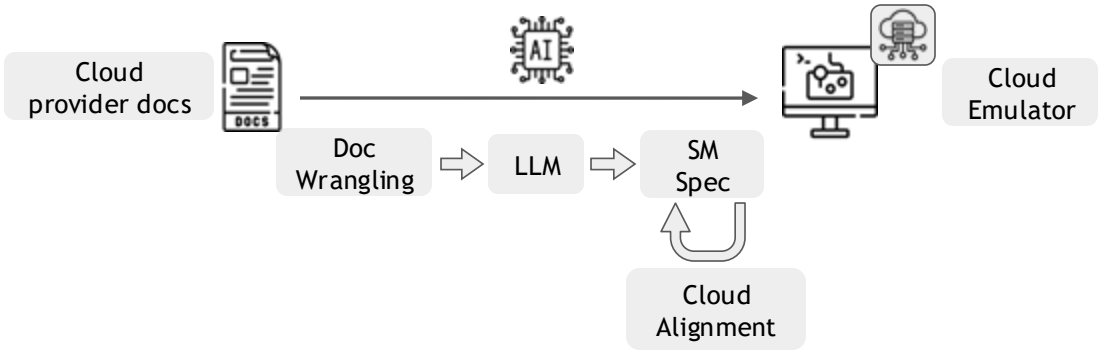
Future Work

Scaling up, Multi-cloud

Cloud Gym: No-cost and no-risk gym environments for cloud agents, AI-assisted CloudOps



Questions?



Check out the paper!

Looking for internships for Summer 2026!

Email – architb@umich.edu