India
Timezone: Indian Standard Time (UTC+5:30)

Project Idea for Volcano

LFX Project proposal for Volcano

Personal Details and Contact Information

Full Name- **Ronak Raj**
Github username - https://github.com/RONAK-AI647
Email - codeitronak226277@gmail.com

Synopsis

- The Volcano JobFlow feature enables users to orchestrate complex batch workflows in Kubernetes. This project aims to enhance JobFlow by introducing:
  ✅ **JobTemplate Parameter Overrides** — Allow per-step customization without duplicating templates.
  ✅ **Retry Policies** — Automate retries for failed jobs with configurable strategies.
  ✅ **Advanced Control Flow** — Support `if`, `switch`, and `for` statements for dynamic workflows.
- These enhancements will make JobFlow more flexible, powerful, and production-ready for diverse use cases like AI/ML pipelines, data processing, and HPC workloads.

Benefits to the Community

By enhancing JobFlow, this project will:

- **Reduce the need for duplicate JobTemplates**, improving maintainability and reducing management overhead.
- Enable **resilient job execution** with automated retry logic, leading to better reliability in large-scale, multi-step workflows.
- Allow users to create **conditional and dynamic workflows**, unlocking advanced use cases like branching execution paths and iterative tasks.
- These improvements will position Volcano as a **more robust and production-ready solution** for complex batch processing in Kubernetes environments

Deliverables

Deliverable	Description	Timeline
JobTemplate Parameter Overrides	Enable per-step customization of JobTemplate fields (resources, env vars, etc.) within a JobFlow.	4 weeks
Retry & Backoff Policies	Add `maxRetries`, `retryPolicy`, and `backoffPolicy` fields in JobFlow spec; implement controller logic for retries.	4 weeks
Advanced Control Flow Statements	Support `if`, `switch`, and `for` constructs in JobFlow spec and execution logic.	4 weeks
Documentation & Examples	Update Volcano documentation, add YAML samples and tutorials for new JobFlow features.	2 weeks
Testing & Community Feedback	Write unit/integration tests; incorporate community feedback.	Throughout

Approach

1)Allow JobTemplate Parameter Overrides in JobFlow

So in Volcano , if a JobFlow step references a JobTemplate, it uses the JobTemplate as-is—no customization allowed.So here we try allow users to override certain parameters (like resource limits, environment variables, commands, etc.) within each JobFlow step, *without modifying the original JobTemplate*.
.This makes Jobflow more reusable and flexible. We will deal iot like this

- [ ] **Add API for overrides.**
- [ ] **Apply overrides when creating Jobs.**
- [ ] **Minor adjustments if needed.**
- [ ] **Document the new feature.**
- [ ] **Write tests.**

2)Implement Job Failure Retry Mechanism: Implement a configurable retry policy for jobs within a JobFlow.

- [ ] **Add retry API**
- [ ] **Implement logic**
- [ ] **Adjust state machine**
- [ ] **Documentation**
- [ ] **Tests**

The approach for adding Advanced Control Flow Statements (if, switch, for) to JobFlow.

- [ ] **Extend API Spec: Add support for `if`, `switch`, and `for` in the `JobFlow` YAML**
- [ ] **Update Controller Logic: Parse conditions, track state, decide which steps to run**
- [ ] **Add Evaluation Engine: Code that evaluates conditions and variables at runtime**
- [ ] **Update YAML Parser: Accept new fields in JobFlow spec**
- [ ] **Update Docs: Add usage examples**
- [ ] **Write Tests: Cover `if`, `switch`, `for`**

Expected Outcomes

- **A fully enhanced JobFlow API supporting:**
    - **Parameter overrides**
    - **Retry logic**
    - **Advanced control flow (`if`, `switch`, `for`)**
- **Updated CRDs and controller logic in Volcano.**
- **Comprehensive tests and documentation.**
- **Demo workflows showcasing new features.**

File	Purpose
controllers/apis	Define new fields in JobFlow CRD (e.g., parameterOverrides, retryPolicy, control flow fields).
controllers/jobflow/jobflow_controller.go	Implement core logic for parameter overrides, retries, and control flow handling.
controllers/jobflow/state/	Update state management for retry handling and dynamic execution.
`utils/flowlogic.go` (new)	Implement condition evaluation and control flow utilities.

`test/e2e/jobflow/`	**Add test cases for all new features.**
`docs/design/jobflow/README.md`	**Update documentation with examples and usage guides.**

- # Timeline

Period	Task
Community Bonding (Week 1–2)	**Familiarize with the Volcano codebase, Kubernetes controllers, and current JobFlow implementation. Interact with mentors and**
Phase 1 (Week 3–6) **Phase 2 (Week 7–10)**	**Implement JobTemplate Overrides; draft PR for community feedback.** **Implement Retry and Backoff Policies; add tests and update documentation.**
Phase 3 (Week 11–14) **Final Submission (Week 15–16)**	**..Implement Advanced Control Flow (if/switch/for); ensure stability and test coverage.** **Complete documentation, final testing, and showcase results to the community.**

Feature Implementation Plan

1) Parameter Overrides:

- **Add parameterOverrides field in JobFlowStepSpec to allow overriding resource limits, commands, etc.**
- **Modify controller logic to apply overrides at runtime.**

2) Retry Mechanism:

- **Add fields: maxRetries, retryPolicy, backoffPolicy in JobFlowStepSpec.**
- **Implement retry logic in controller: track job status, retry based on policy, and update status accordingly.**

3) Control Flow Statements:

- **Add support for if, switch, for in JobFlowStepSpec.**
- **Create an evaluation engine to process conditions (preprocessing.status == 'Success') and variables.**
- **Dynamically generate/skip steps based on evaluation.**

4) Testing and Documentation:

- **Write e2e tests for various workflows.**
- **Add YAML examples and usage in documentation.**
    -

Why me?

I am deeply passionate about Kubernetes, cloud-native systems, and open-source development. My technical background in Golang, distributed systems, and Kubernetes architecture aligns perfectly with this project. I have already explored the Volcano codebase, participated in relevant discussions, and demonstrated a strong understanding of scheduler internals, plugin mechanisms, and simulation frameworks.

I am a fast learner and a dedicated contributor, eager to collaborate with the community, learn from mentors, and deliver impactful solutions. I am driven by the desire to make complex systems like Volcano more transparent and accessible for developers, researchers, and users. Given the opportunity, I will ensure this project meets high-quality standards, with clear documentation, tests, and a user-focused approach.

I truly believe that dedicated new contributors deserve opportunities in open source, especially when they're ready to commit fully and treat the project as a primary focus rather than a side task — something that aligns with the values LFX promotes. I currently have several open issues and pull requests in the Volcano Dashboard repository, and I'm fully committed to completing them and contributing my best to the project.

ISSUES:

- https://github.com/volcano-sh/dashboard/issues/146
- https://github.com/volcano-sh/dashboard/issues/127
- https://github.com/volcano-sh/dashboard/issues/124

- https://github.com/volcano-sh/dashboard/issues/120
- https://github.com/volcano-sh/dashboard/issues/119
- https://github.com/volcano-sh/dashboard/issues/90
- https://github.com/volcano-sh/dashboard/issues/8

PULL REQUESTS:

- https://github.com/volcano-sh/dashboard/pull/137**(prepare DCO sign off)**
- https://github.com/volcano-sh/dashboard/pull/136**(create docker-compose.yml)**
- https://github.com/volcano-sh/dashboard/pull/126**(husky set up)**
- https://github.com/volcano-sh/dashboard/pull/125**(pull request template)**

-