**PART 1: Set Up a GitLab Runner on a VM (Self-hosted, Docker executor)**

**✅ Prerequisites**

* Ubuntu 20.04 VM with:
  + Docker installed (docker info should work)
  + OpenSSH access
  + Internet access
* GitLab repo with access

**📥 Step 1: Install GitLab Runner**

# Add GitLab runner repository

curl -L https://packages.gitlab.com/install/repositories/runner/gitlab-runner/script.deb.sh | sudo bash

# Install GitLab Runner

sudo apt-get install gitlab-runner -y

**⚙️ Step 2: Register the Runner**

1. On GitLab, go to your **project** or **group**:
   * Settings > CI/CD > Runners > Expand
2. Copy the **registration token**.
3. On your VM, run:

sudo gitlab-runner register

**You'll be prompted:**

* **GitLab instance URL:**
* https://gitlab.com
* **Registration token:**  
  *(Paste from GitLab)*
* **Description:**  
  vm-runner-01
* **Tags:**  
  docker,deploy
* **Executor:**  
  docker
* **Docker image:**
* docker:20.10

**🐳 Step 3: Enable Docker-in-Docker (DinD)**

Edit /etc/gitlab-runner/config.toml:

[runners.docker]

tls\_verify = false

image = "docker:20.10"

privileged = true # <---- important for DinD

volumes = ["/cache", "/var/run/docker.sock:/var/run/docker.sock"]

⚠️ privileged = true is required to allow Docker builds inside the runner.

Then restart:

sudo gitlab-runner restart

**✅ Done! Your runner should now be online and ready for CI/CD jobs.**

**🔁 PART 2: Enable Automatic Rollback on Deployment Failure**

**💡 Strategy:**

1. Back up the current docker-compose state
2. Deploy new version
3. If the deploy fails, roll back to the previous version

**🧠 Modify your deploy job in .gitlab-ci.yml:**

deploy:

stage: deploy

script:

- |

ssh $DEPLOY\_USER@$DEPLOY\_HOST << 'EOF'

set -e

cd $APP\_DIR

echo "Creating backup of current docker-compose..."

docker-compose ps -q > .prev\_containers || true

cp docker-compose.yml docker-compose.prev.yml || true

echo "Deploying new version..."

if docker-compose pull && docker-compose up --build -d; then

echo "✅ Deploy successful"

else

echo "❌ Deploy failed, rolling back..."

if [ -f docker-compose.prev.yml ]; then

mv docker-compose.prev.yml docker-compose.yml

docker-compose up -d

fi

# Optionally, remove failed containers

if [ -s .prev\_containers ]; then

xargs docker rm -f < .prev\_containers

fi

exit 1

fi

EOF

**🔒 Why this works:**

* Uses set -e to exit on any error
* Keeps .prev\_containers list and backup of docker-compose.yml
* Falls back to previous version if docker-compose up fails

**📌 Optional: Use Git Commit Tagging for Rollbacks**

Instead of backing up the compose file, you can use Git tags or branches to track stable versions.

Would you like a **Blue/Green deployment** setup instead for zero-downtime rollbacks?

Let me know if you want:

* A systemd service to auto-start the app on reboot
* Auto-notification to Slack/Discord on deploy
* HTTPS/SSL setup for production

Happy to help you scale it securely.