1. Optimize Pipelines & Agent Usage

- **Use Microsoft-hosted agents only when needed**
    - They charge per minute after free tier (1,800 minutes for public projects, 2,000 for private).
    - Consider self-hosted agents if you have long-running jobs.
- **Use Job Timeouts**: Set timeouts for jobs to avoid long-running pipelines that consume unnecessary resources.
- **Use Self-Hosted Agents:** If you have consistent pipeline workloads, consider using self-hosted agents instead of Microsoft-hosted agents. This can significantly reduce costs, especially for large teams or frequent builds.
- **Scale Agents Dynamically**: Use tools like Azure Virtual Machine Scale Sets or Kubernetes to scale self-hosted agents up and down based on demand.
- **Turn off unused self-hosted agents**
    - If you're using Azure VMs for agents, shut them down when not in use.
- **Optimize pipeline steps**
    - Reduce redundant builds and tests.
    - Use caching (Cache task) to speed up workflows.
    - Run jobs in parallel only if necessary.

2. Manage Artifact Storage Costs

- **Use retention policies**
    - Limit how long artifacts, logs, and pipeline runs are stored.
    - Default is unlimited—adjust it in project settings.
- **Move artifacts to cheaper storage**
    - Instead of Azure Artifacts, consider Azure Blob Storage with lifecycle policies.
- **Delete old releases**
    - Set retention rules to clean up unused release pipelines and environments.

3. Optimize Repository & Code Storage

- **Use shallow clones in pipelines**
  - Reduce repo size by pulling only necessary commits:

    yaml

    CopyEdit

    ```
    steps:
      - checkout: self
        fetchDepth: 1  # Fetch only the latest commit
    ```
- **Monitor large repositories**
  - Use Git LFS for large binary files instead of bloating your repo.
- **Limit branch policies**
  - Reduce excessive builds by using trigger filters on important branches only.

4. Control User Licensing & Access

- **Review paid user assignments**
  - Use the **"Stakeholder"** role (which is free) for users who don't need advanced features.
- **Check unused licenses**
  - Remove or downgrade users who don’t need full Azure DevOps access.
- **Use group-based access control**
  - Assign permissions at the group level instead of per user to simplify management.

5. Optimize Test & Deployment Costs

- **Run tests selectively**
  - Only execute full test suites on major changes.
  - Use test impact analysis to run only necessary tests.
- **Use cheaper environments for testing**
  - Deploy to lower-cost VMs instead of expensive ones.
  - Consider **Azure Spot VMs** for non-production workloads.
- **Auto-scale environments**
  - Use **Azure DevTest Labs** or **Scale Sets** to reduce infrastructure costs.

6. Use Azure Cost Management & Alerts

- **Set up budgets & alerts**
  - Define cost limits in **Azure Cost Management + Billing** to avoid unexpected charges.
- **Analyze cost reports**
  - Regularly review **Cost Analysis** to identify expensive services.
- **Use Azure Reservations**
  - Prepay for services like VMs to get discounts (up to 72% off).

7. Use YAML Pipelines

- **YAML Pipelines**: Migrate from classic pipelines to YAML pipelines for better control over pipeline configuration and resource usage.
- **Reuse Templates**: Create reusable YAML templates to avoid duplicating pipeline code and reduce maintenance overhead.

8. Optimize Licensing

- **Basic vs. Paid Licenses**: Ensure that users who only need basic access (e.g., stakeholders) are assigned the appropriate license type to avoid unnecessary costs.
- **Azure DevOps Server**: If you have a large team and prefer on-premises solutions, consider using Azure DevOps Server instead of the cloud-based service.

9. Review and Optimize Regularly

- **Audit Pipelines**: Regularly review your pipelines to identify inefficiencies or unused resources.
- **Stay Updated**: Keep up with Azure DevOps updates and new features that can help reduce costs.

Summary

Optimization Area	**Cost-Saving Tip**
Pipelines	Use self-hosted agents, optimize jobs, cache dependencies
Artifacts	Set retention policies, move storage to Blob, delete old releases
Repos	Use shallow clones, avoid large files in Git, limit builds
Users	Assign Stakeholder roles, remove unused licenses
Testing	Run tests selectively, use cheaper environments
Monitoring	Set budgets, analyze cost reports, use reservations