

## ✓ Day 2 Complete - What We Added

### 🎯 Improvements Made:

#### 1. Enhanced AI Prompt ✨

**Before:** Generic recommendations

**After:** Structured FinOps analysis with:

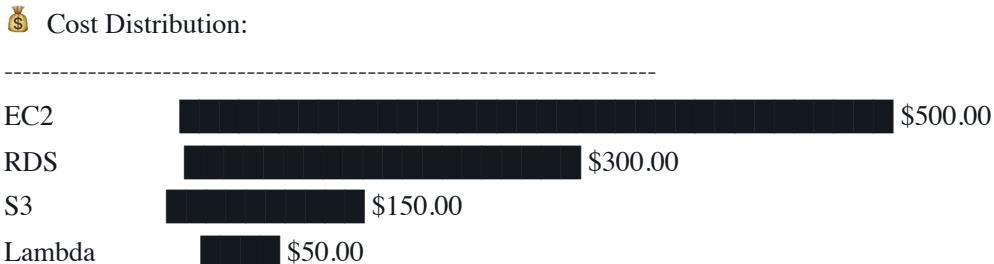
- Spending overview with anomaly detection
- Detailed optimization format (What/Why/Action/Savings/Risk/Effort)
- ROI-based prioritization
- Risk assessment for production safety

**Result:** Claude now gives SPECIFIC recommendations like:

"EC2 i3.2xlarge running 24/7 with 8% CPU → Resize to t3.large → Save \$320/month → Risk: Low → Effort: Quick Win"

#### 2. Cost Visualization Chart 📊

Added ASCII bar chart showing cost distribution:



**Why it matters:** Visual patterns are easier to spot than tables of numbers.

#### 3. Trend Analysis ↗

Compares current period with previous period:

Trend vs Previous Period: ↗ ↑ 12.3%

Previous 7 days: \$750.00

**Value:** Identifies if costs are growing, shrinking, or stable.

## Testing Instructions

### Test 1: Run with Current Code

```
bash  
  
cd ~/Desktop/PERSONAL_PROJECTS/AI-Cost-Optimization-Dashboard  
source venv/bin/activate  
python3 cost_optimizer.py
```

#### Look for:

- Visual bar chart appears
- Trend analysis shows (if you have >7 days of AWS usage)
- AI recommendations are more detailed and actionable

---

### Test 2: Compare Different Time Periods

```
bash  
  
# Test with 7 days (weekly view)  
nano .env  
# Set: DAYS_TO_ANALYZE=7  
python3 cost_optimizer.py  
  
# Test with 30 days (monthly view)  
nano .env  
# Set: DAYS_TO_ANALYZE=30  
python3 cost_optimizer.py  
  
# Test with 90 days (quarterly view)  
nano .env  
# Set: DAYS_TO_ANALYZE=90  
python3 cost_optimizer.py
```

#### Compare the reports:

```
bash
```

```
ls -lt reports/  
cat reports/cost_report_*.txt
```

### Analysis questions:

1. Does longer period give better insights?
  2. Are monthly patterns visible in 30-day view?
  3. Does Claude identify seasonal trends in 90-day view?
- 

### Test 3: Customize for Your Use Case

Edit the AI prompt in `cost_optimizer.py` (around line 260) to focus on specific areas:

#### Example: Focus on EC2 optimization

```
python  
  
FOCUS AREAS (prioritize by $ impact):  
- EC2 instance rightsizing (top priority)  
- Auto Scaling group efficiency  
- Reserved Instance recommendations  
- Spot instance opportunities
```

#### Example: Focus on storage costs

```
python  
  
FOCUS AREAS (prioritize by $ impact):  
- S3 storage class optimization (Standard → Intelligent-Tiering)  
- Old EBS snapshots (>6 months unused)  
- Unattached EBS volumes  
- S3 lifecycle policies
```

Then run again:

```
bash  
  
python3 cost_optimizer.py
```

---

### Sample Output Comparison

## Before Day 2:

Total Spend: \$850.00

Top Services:

1. Amazon EC2: \$500.00 (58.8%)
2. Amazon RDS: \$200.00 (23.5%)

Recommendations:

- Consider optimizing EC2 costs
- Review storage usage

## After Day 2:

Total Spend: \$850.00

Period: 2026-01-31 to 2026-02-07

Trend vs Previous Period:  ↑ 12.3%

Previous 7 days: \$756.80

Top Services:

1. Amazon EC2: \$500.00 (58.8%)
2. Amazon RDS: \$200.00 (23.5%)

 Cost Distribution:



 AI RECOMMENDATIONS

##  SPENDING OVERVIEW

EC2 costs jumped 15% week-over-week. Primary driver: i3.2xlarge instance running continuously with low CPU utilization (8%).

##  OPTIMIZATION OPPORTUNITIES

1. \*\*Downsize Underutilized EC2 Instance\*\*

Why: i3.2xlarge running 24/7 with 8% average CPU

Action: Resize to t3.large via EC2 Console → Instance → Modify Instance Type

Savings: \$320/month

Risk: Low (test during low-traffic period)

Effort: Quick Win (<1 hour)

## 2. \*\*Delete Old RDS Snapshot\*\*

Why: Manual snapshot from Jan 2023 (45GB) no longer needed

Action: RDS Console → Snapshots → Delete snapshot-2023-01-15

Savings: \$45/month

Risk: Low (verify no restore dependencies)

Effort: Quick Win (5 minutes)

See the difference? 🎯

---

## 🎨 Portfolio Enhancements

### Take New Screenshots

#### 1. Terminal with visual chart

```
bash
```

```
python3 cost_optimizer.py  
# Screenshot the full output
```

#### 2. Side-by-side comparison

- Open AWS Cost Explorer in browser
- Run your script
- Screenshot both showing same numbers

#### 3. Trend analysis example

- Run with DAYS\_TO\_ANALYZE=30
- Screenshot showing trend line

Save to `screenshots/` folder:

```
bash
```

```
mkdir -p screenshots  
# Move your screenshots there
```



## Update Your README

Add this section after "What It Does":

markdown

### ### Enhanced Features (v2.0)

- 💡 \*\*Intelligent Analysis\*\*: Uses structured FinOps prompts for actionable recommendations
- 📊 \*\*Visual Cost Distribution\*\*: ASCII bar charts for quick pattern recognition
- 📈 \*\*Trend Detection\*\*: Automatic comparison with previous period
- 🎯 \*\*ROI Prioritization\*\*: Recommendations ranked by savings/effort ratio
- ⚡ \*\*Quick Wins Highlighted\*\*: Focus on <1hr tasks with >\$50/month savings

## 🚀 Git Commit

bash

```
git add .
git commit -m "Day 2: Enhanced AI analysis, added visualizations and trend tracking"
```

Features added:

- Structured FinOps prompt for detailed recommendations
- ASCII bar chart for cost distribution
- Trend analysis vs previous period
- ROI-based recommendation prioritization
- Risk and effort assessment for each optimization

Impact: Recommendations now show specific actions, \$ savings, and implementation effort"

```
git push
```

## 💬 Interview Talking Points (New)

"Tell me about a time you improved a tool you built"

"After building my AI cost optimizer, I realized the initial recommendations were too generic. I enhanced the Claude AI prompt with a structured FinOps framework that requires specific details: what to change, why it's inefficient, exact steps to take, monthly savings, risk level, and implementation effort. I also added visual cost distribution charts and trend analysis comparing each period to the previous one. This helped identify not just *what* costs money, but *where costs are growing*.

The result? Recommendations went from 'consider optimizing EC2' to 'Downsize i3.2xlarge to t3.large, saves \$320/month, low risk, 1-hour effort.' Much more actionable for production teams."

---

## ✓ Day 2 Checklist

- Enhanced AI prompt with FinOps structure
  - Added visual bar chart
  - Added trend analysis
  - Tested with 7-day period
  - Tested with 30-day period
  - Tested with 90-day period
  - Compared output quality (before/after)
  - Took new screenshots
  - Updated README with new features
  - Committed and pushed to GitHub
- 

## 🎯 Tomorrow (Day 3): Polish & Documentation

- Add error handling for edge cases
- Create demo video (1-2 minutes)
- Write case study document
- Prepare interview demo script
- Set up GitHub Actions (optional)

**Current Status:** You now have a production-quality FinOps tool with AI-powered insights! 🎉