



# Complete Version Strategy & Reference Guide

## Version Roadmap Overview

### ✓ Version 1.x - Foundation (MVP → Polish → User Feedback)

- **1.0.0** - Basic MVP (2 weeks) - *See Artifact 02-NEW*
- **1.1.0** - Polish & Templates (1 week) - *See Artifact 03-NEW*
- **1.2.0** - User Requested Features (2 weeks)
- **1.3.0** - Payment Integration (1 week)



### Version 2.x - Scale (Performance → Features → Optimization)

- **2.0.0** - Infrastructure Upgrade (1 month)
- **2.1.0** - Multi-LLM Support (2 weeks)
- **2.2.0** - Advanced Analytics (2 weeks)
- **2.3.0** - File Export Features (1 week)



### Version 3.x - Enterprise (Teams → API → White-label)

- **3.0.0** - Team Collaboration (1 month)
- **3.1.0** - API Platform (2 weeks)
- **3.2.0** - Custom Domains (1 week)
- **3.3.0** - White-label Solution (2 weeks)

## When to Reference Original Artifacts

### For Version 1.2.0 Features:

- **Payment Processing** → See Artifact 19 (Financial Infrastructure)
- **SSL Certificates** → See Artifact 14 (Critical Additions)
- **Better Deployment** → See Artifact 15 (Prompt Builder)

### For Version 2.0.0 Infrastructure:

- **Redis/Caching** → See Artifact 14 (Critical Additions)
- **Job Queues** → See Artifact 20 (Final Review)
- **Database Optimization** → See Artifact 02 (Original Schema)

## For Version 2.1.0 Multi-LLM:

- **Complete Implementation** → See Artifact 19 (Multi-LLM section)
- **Cost Optimization** → See Artifact 19 (Financial AI)

## For Version 3.0.0 Enterprise:

- **Full Architecture** → See Artifact 01 (Complete Vision)
- **Security Hardening** → See Artifact 20 (Security section)
- **Custom Domains** → See Artifact 20 (Enhancement Ideas)

## Version 1.2.0 - Quick Implementation Guide

### What to Build (2 weeks):

#### 1. Stripe Payment Integration

```
bash

claude "Create payment routes with Stripe for:
- $9/month Pro plan (no ads, 100 tools)
- $29/month Business plan (custom branding, 1000 tools)"
```

#### 2. User Dashboard

```
bash

claude "Create user dashboard showing:
- Tools created
- Total usage
- Revenue (if ads enabled)
- Subscription status"
```

#### 3. SSL for Tools

```
bash

# Simple approach for MVP
claude "Create a script that:
- Generates Let's Encrypt cert for each tool subdomain
- Updates nginx config
- Auto-renews certificates"
```

#### 4. Basic Versioning

```
bash
```

claude "Add version tracking to prompts table  
Allow reverting to previous versions"

## Version 1.3.0 - Payment Features (1 week)

### Simple Subscription System:

```
sql

ALTER TABLE users ADD COLUMN
subscription_tier VARCHAR(20) DEFAULT 'free',
subscription_expires TIMESTAMP,
stripe_customer_id VARCHAR(255);
```

### Usage Limits:

```
javascript

const TIER_LIMITS = {
  free: { tools: 3, uses_per_day: 100 },
  pro: { tools: 100, uses_per_day: 10000 },
  business: { tools: 1000, uses_per_day: 100000 }
};
```

## Version 2.0.0 - Infrastructure Upgrade

### When You Need It:

- Server CPU consistently > 60%
- Database queries taking > 100ms
- Users complaining about speed
- Making > \$5000/month

### What to Add:

#### 1. Redis Caching

```
bash

# Reference Artifact 14 for implementation
# Cache: sessions, Claude responses, tool configs
```

#### 2. Background Jobs

```
bash
```

```
# Reference Artifact 20 for Bull queue setup
```

```
# Queue: deployments, emails, analytics
```

### 3. CDN for Tools

```
bash
```





```
# Use Cloudflare for static assets
```

```
# Reduces server load by 70%
```

## Claude Code Usage by Version





### Version 1.x (Maximum Claude Usage)

Use Claude for:

-  All CRUD operations
-  Basic integrations
-  UI components
-  Simple features





### Version 2.x (Balanced Approach)

Use Claude for:

-  Boilerplate code
-  API integrations
-  Complex features (with review)
-  Architecture decisions

### Version 3.x (Strategic Usage)

Use Claude for:

-  Documentation
-  Test writing
-  Feature implementation (with specs)
-  Security-critical code

# Migration Strategy Between Versions

## Before Each Major Version:

1. **Full database backup**
2. **User notification** (1 week advance)
3. **Feature freeze** (3 days before)
4. **Staging environment test**

## Rollback Plan Template:

```
bash

#!/bin/bash
# Always have this ready

# 1. Stop services
pm2 stop all

# 2. Restore database
psql -h $DB_HOST -U $DB_USER -d $DB_NAME < backup-pre-$VERSION.sql

# 3. Restore code
git checkout v$PREVIOUS_VERSION

# 4. Restart
pm2 restart all
```

## Revenue Milestones

### By Version Expected Revenue:

- **v1.0.0:** \$100/month (ads only)
- **v1.1.0:** \$500/month (better retention)
- **v1.2.0:** \$1000/month (some paying users)
- **v1.3.0:** \$2000/month (subscriptions)
- **v2.0.0:** \$5000/month (scale)
- **v2.x:** \$10000/month (optimization)
- **v3.x:** \$20000+/month (enterprise)

## Decision Framework

## When to Move to Next Version:

1. Current version is stable (< 5 bugs/week)
2. Users are asking for next features
3. Revenue justifies development time
4. You have time to support it

## When to Skip Features:

1. < 10% of users would use it
2. Adds > 20% to codebase complexity
3. Cheaper alternative exists
4. Delays core features by > 1 week

## The Golden Rules

1. **Ship v1.0.0 in 2 weeks** - No excuses
2. **Let users drive features** - Don't guess
3. **Revenue before perfection** - Make money first
4. **Use Claude Code aggressively** - Speed matters
5. **Keep versions small** - Easier to debug
6. **Always have rollback plan** - Things break
7. **Document what worked** - For next time

## Your Immediate Action Items

1. Start with Artifact 02-NEW (Version 1.0.0 MVP)
2. Set up basic monitoring (even just uptime robot)
3. Create a feedback form
4. Launch in 2 weeks
5. Celebrate! 🎉

Remember: Version 1.0.0 doesn't need to be perfect. It needs to exist!