Dashblock Development Roadmap

Minecraft Server Management Platform

Executive Summary

Project Vision: Simplify Minecraft server deployment and management by providing a web-based dashboard that automates the complex process of setting up and managing Minecraft servers on Oracle Cloud (and other VPS providers).

Target Users: Minecraft players who want to create servers without technical complexity

Core Value Proposition: Transform the manual process of SSH connection, file uploads, command execution, and server configuration into a simple web interface.

Architecture Decisions

Backend Framework: FastAPI

Rationale:

- **Async Support:** Essential for handling multiple concurrent SSH connections
- **High Performance:** Critical for real-time server operations and monitoring
- **Automatic Documentation:** Accelerates API development and testing
- **Type Safety:** Pydantic validation prevents common errors
- **Modern Architecture:** Future-proof technology stack

SSH Security Model: Temporary Key Upload

Implementation:

- Users upload private SSH keys through secure form
- Keys stored encrypted in Redis with 30-60 minute TTL
- Ed25519 key format recommended (superior security to RSA)
- Secure memory handling during SSH operations
- Automatic cleanup and rotation

Architecture Pattern: Direct SSH Connection

Benefits:

- No agent installation required on user VPS
- Simpler security model with fewer attack vectors
- User maintains full control of their infrastructure
- Better customer isolation
- Reduced maintenance overhead

Development Phases

Phase 1: Core Infrastructure (2-3 weeks)

Objectives: Establish foundation and basic VPS connectivity

Week 1: Foundation Setup

- FastAPI project structure with async architecture
- User authentication system (JWT-based)
- Database schema design (PostgreSQL + SQLAlchemy)
- Redis setup for SSH key caching
- Basic frontend structure (HTML/CSS/JavaScript)

Week 2: SSH Integration

- SSH key upload and validation system
- Secure key storage in Redis with encryption
- SSH connection testing and validation
- Basic VPS connectivity interface
- Error handling and logging system

Week 3: Dashboard Core

- VPS management interface
- Server listing and status display
- Connection status monitoring
- Basic security measures implementation

Deliverables:

- ✅ Working VPS connection system
- ✅ SSH key management
- ✅ Basic dashboard interface
- ✅ User authentication

Phase 2: Server Management Foundation (3-4 weeks)

Objectives: Implement core Minecraft server operations

Week 4-5: Process Management

- Tmux session creation and management
- Server startup/shutdown controls
- Process monitoring and status tracking
- Command execution interface

Week 6: File Management

- Secure file upload system (server.jar, configs)
- Directory structure creation
- EULA acceptance automation
- Server properties management (online-mode toggle)

Week 7: Server Configuration

- Server runner detection (Fabric/Forge)
- Java version management
- Memory allocation controls
- Port configuration and validation

Deliverables:

- ✅ Complete server lifecycle management
- ✅ File upload and configuration
- ✅ Automated EULA handling
- ✅ Basic server controls

Phase 3: Modpack Integration (4-5 weeks)

Objectives: Automate modpack installation from major platforms

Week 8-9: API Integration

- CurseForge API integration and authentication
- Modrinth API integration
- Modpack search and browse interface
- Dependency resolution system

Week 10-11: Installation System

- Automated modpack download
- Server type detection (Fabric/Forge/Quilt)
- Mod compatibility checking
- Installation progress tracking

Week 12: Advanced Features

- Custom modpack upload support
- Version management and updates
- Rollback capabilities
- Configuration synchronization

Deliverables:

- ✅ CurseForge integration
- ✅ Modrinth integration
- ✅ Automated modpack installation
- ✅ Dependency management

Phase 4: Advanced Features (3-4 weeks)

Objectives: Add monitoring, management, and multi-server support

Week 13: Server Monitoring

- Real-time CPU/RAM monitoring
- Player count and status tracking
- Log file streaming and search
- Performance metrics dashboard

Week 14: Player Management

- Player list and status
- Whitelist/blacklist management
- Permission system integration
- Ban/kick functionality

Week 15: Backup & Multi-Server

- Automated backup scheduling
- World download capabilities
- Multi-server support per VPS
- Server cloning and templates

Deliverables:

- ✅ Complete monitoring system
- ✅ Player management tools
- ✅ Backup automation
- ✅ Multi-server capabilities

Phase 5: Polish & MVP (2-3 weeks)

Objectives: Prepare for production deployment

Week 16: Quality Assurance

- Comprehensive error handling
- Input validation and sanitization
- Security audit and hardening
- Performance optimization

Week 17: User Experience

- UI/UX improvements and polish
- Mobile responsiveness
- User documentation and tutorials
- Help system and troubleshooting guides

Week 18: Deployment Preparation

- Production deployment configuration
- Monitoring and alerting setup
- Backup and disaster recovery
- Load testing and optimization

Deliverables:

- ✅ Production-ready application
- ✅ Complete documentation
- ✅ Security hardening
- ✅ Performance optimization

Technical Specifications

Technology Stack

- **Backend:** FastAPI + Python 3.11+
- **Database:** PostgreSQL (main data) + Redis (cache/sessions)
- **Frontend:** HTML5, CSS3, JavaScript (ES6+)
- **Infrastructure:** Docker containers for easy deployment
- **Monitoring:** Prometheus + Grafana for metrics

Security Features

- JWT-based authentication with refresh tokens
- SSH key encryption and automatic expiration
- Input validation and sanitization
- Rate limiting and DDoS protection
- Audit logging for all operations

Performance Targets

- Dashboard load time: <2 seconds
- SSH connection establishment: <5 seconds
- Modpack installation: Progress tracking with ETA
- Support for 100+ concurrent users per instance

Scalability Considerations

Phase 6+: Future Enhancements

- **Multiple VPS Provider Support:** AWS, DigitalOcean, Vultr
- **Advanced Monitoring:** Custom metrics and alerting
- **Plugin Ecosystem:** Custom mod/plugin management
- **Collaborative Features:** Team server management
- **API Access:** Third-party integrations
- **Enterprise Features:** Organization management, billing

Infrastructure Scaling

- Microservices architecture for components
- Load balancing for high availability
- Database sharding for large user bases
- CDN integration for file distributions

Success Metrics

MVP Success Criteria

- Successfully deploy and manage 10+ different modpacks
- Handle 50+ concurrent server management sessions
- 99.9% uptime for dashboard operations
- Average server setup time: <10 minutes
- User satisfaction: 4.5+/5.0 rating

Technical KPIs

- SSH connection success rate: >99%
- Modpack installation success rate: >95%
- Average response time: <500ms
- Error rate: <1%

Risk Mitigation

Technical Risks

- **SSH Connection Failures:** Implement retry logic and connection pooling
- **VPS Provider API Changes:** Abstract API layers for easy adaptation
- **Security Vulnerabilities:** Regular security audits and updates
- **Performance Issues:** Comprehensive monitoring and optimization

Business Risks

- **Market Competition:** Focus on user experience and automation
- **Scalability Challenges:** Design for horizontal scaling from start
- **User Adoption:** Comprehensive documentation and support

Resource Requirements

Development Team (Recommended)

- 1 Backend Developer (FastAPI/Python)
- 1 Frontend Developer (HTML/CSS/JavaScript)
- 1 DevOps Engineer (part-time)
- 1 QA Tester (part-time)

Infrastructure Costs (MVP)

- Development server: $20-50/month
- Database hosting: $25-100/month
- CDN and assets: $10-30/month
- Monitoring tools: $0-50/month
- **Total Monthly:** $55-230/month

Conclusion

This roadmap provides a comprehensive path to building Dashblock as a production-ready Minecraft server management platform. The 18-week timeline balances feature completeness with rapid iteration, allowing for user feedback integration throughout the development process.

The technical architecture choices (FastAPI, direct SSH, temporary key handling) optimize for performance, security, and user experience while maintaining simplicity and scalability.

Next Steps:

1. Begin Phase 1 implementation
2. Set up development environment and CI/CD pipeline
3. Create detailed technical specifications for each phase
4. Establish user feedback channels for iterative improvements