Table of Contents

Preface ..... **xi**

1. The Puppet Design Philosophy **1**
Declarative Code 1
What Is Declarative Code, Anyway? 2
Resource Types and Providers 3
Procedural Example 4
Nondeclarative Code with Puppet 8
Idempotency 10
Side Effects 11
Resource-Level Idempotence 12
Run-Level Idempotence 14
Nondeterministic Code 15
Stateless 16
Benefits of Stateless Design 17
Sources of State Information 18
Summary 20

2. High-Level Code and Data Design **21**
Code and Data Organization 21
Code and Data Categories 22
Types of Code Logic 23
Examples of Logic Types 25
Mapping Data Types to Puppet Use Cases 26
Application Logic and Puppet Modules 26
Business Logic Should Not Be Written into Component Modules 29
Business Logic with Roles and Profiles 30
Business, Service, Site, Node, and Application Data 31

Hiera Data Sources 32
Node Classification 33
Summary 35

3. Coding Practices. 37
The Style Guide 37
Coding Principles 38
KISS: Keep It Simple 39
The Single Responsibility Principle 40
Separation of Concerns 41
Interface-Driven Design 42
DRY: Don't Repeat Yourself 44
Don't Reinvent the Wheel 48
Code Practices 48
Balance of Resources Versus Logic 49
Balance of Code Versus Data 49
Conditional Logic 50
Iteration 52
Generating Lists 54
Variables 55
Variable Naming 55
Referencing Variables 55
Other Variable Use Cases 58
Trusted Variables 59
Order of Assignment for Top-Level Variables 60
Assignment with Selectors 60
Attribute Values Chosen by Conditional Evaluation 60
Variable Inheritance 61
Strict Variables 62
Function Calls 63
Always Use Parentheses 63
Functions for Logging and Debugging 63
String Manipulation Functions 64
Path Manipulation 64
Input Validation Functions 64
Catalog Tests 66
Data Transformation 68
Templates 70
ERB Templates 70
EPP Templates 71
EPP Versus ERB 72
Other Language Features 72

Summary 72

4. Puppet Module Design. 75
The Puppet Development Kit 75
Installing the Puppet Agent 76
Using the Ruby that Comes Bundled with Puppet 76
Installing the Puppet Development Kit 76
Favor Editors or IDEs with Puppet Plugins 77
Using Vendor-Provided or Community Modules 77
Picking Good Modules 78
Module Checklist 78
Module Applicability to Your Needs 79
Contributing Modules 81
Designing Modules Well 81
Make Use of Module Structure 81
Keep the Module Focused 81
Design Modules for Public Consumption 82
Planning and Scoping Your Module 82
Basic Module Layout 83
The Module's Main Class 84
Module Parameters 86
Input Validation 90
Data in the Module 91
The params.pp Pattern 91
Hiera Data in Modules 93
Modularizing Classes 95
Dependencies 95
Class Relationships 95
Class Containment 96
Interfacing with Classes 101
Reusing Defined Types 103
Providing Clean Service Interfaces with Defined Types 104
Simplify Complex Operations with a Defined Type 105
Interacting with Other Resouces in the Module 106
Creating Useful Documentation 108
README, REFERENCE, and Other Markdown 108
REFERENCE Markdown 110
Summary 110

5. Resources. 113
Using Resources to Implement Change 113
Resource Types Abstract Implementation Details 114

Use the Most Specific Resource Type 114
Examining a Naked Resource 115
Exploring Resources with Tools 116
Resource Declaration 120
Ensure states 120
Use Variables for Data-Driven Declaration 121
Use Arrays for Similar Resources 122
Using Automatic Resource Relationships for Clean Code 123
Resource Declaration by Functions 124
Resources Metatypes 126
Resource Metaparameters 128
Avoid Parse-Order Problems by Using Virtual Resources 133
Exporting Resources 136
Overriding and Modifying Declared Resources 138
Resource Default Statements 139
Resource Chaining 141
Resource Collectors 141
Resource Best Practices 142
Custom Resource Types 142
Useful Defined Types 147
Summary 148

6. Hiera Data. 151
Separating Code and Data 151
Global, Environment, and Module Data 152
Hiera Backends 153
Designing the Hiera Hierarchy 153
Variable Interpolation 154
Design Guidelines 157
Useful Hierarchy Levels 159
Eliminating Data 161
Accessing Hiera 162
Automatic Parameter Lookups 162
Hiera Function Calls 163
Converting Serialized Hiera Data into Resource Declarations 165
Interpolation in Your Data 166
The Built-In Backends 166
YAML 166
JSON 167
HOCON 168
eYAML 168
Custom Hiera Backends 169

Database and NoSQL Engines 169
Service Discovery Backends 170
Encrypted Key/Value Storage 170
Summary 172

7. Roles and Profiles. 173
Roles 174
Creating Readable Roles 174
Design Roles for a Singular Use Case 176
Provide the Role for Use in Data Lookups 177
Profiles 178
A Sample Service Profile 179
Providing Actionable Data in Profile Parameters 180
Implementing Business Logic in Profiles 181
Defining Module Relationships in Profiles 183
Creating Metaprofiles to Group Configurations 183
Designing an Appropriate Profile Structure 185
Testing Roles and Profiles 187
Validating Profiles by Using Unit Tests 187
Confirming Profile Implementation with Acceptance Tests 188
Summary 189

8. Node Classification. 191
What Data Should the Node Classifier Provide? 191
Roles and Profiles 191
Node-Specific Data 192
Node Statements 192
Node Statement Matching 193
Replacing Node Inheritance 195
Node Parameters Within Node Blocks 197
Fact-Based Classification 198
Fact-Based Role Assignment 198
Security and Fact-Selected Roles or Profiles 198
Fact-Based Hiera Classification 199
Node Parameters with Hiera Classification 200
Avoiding Node Data in Manifests 201
Serverless Classification 202
ENCs 202
What Data Can an ENC Provide? 203
Puppet Management Consoles 204
Inventory and Infrastructure Management ENCs 205
Summary 207

9. Release Engineering and r10k. 209
Puppet Environments in Depth 210
Puppet Directory Environments 210
Selectable Blocks for Catalog Building 210
Environment Configuration 211
Environment Independence and Isolation 211
Deploying with r10k 212
What Does r10k Actually Do? 212
The Control Repository 213
Control Repository Branch Contents 213
r10k Configuration File 215
Puppetfile 215
r10k Deployment Walkthrough 216
Uses for r10k 219
Build Development Environments 219
Simplifying Acceptance Testing 220
Implement Continuous Integration, Delivery, and Deployment 221
Deploy Production Environments 222
Build and Package 223
Release Management Strategies with r10k 224
Stage/Production Branches 225
Single Branch (GitHub Flow) 226
GitFlow 226
Invoking r10k 228
Puppet Prerun Command 228
Deploying on Receipt of a WebHook 229
Orchestrating Deployments with MCollective/Choria 229
Invoking r10k in Testing Frameworks 230
Combining Multiple Invocation Methods 230
Migrating to r10k 231
Repository-per-Module Benefits 231
Configuring an Environment in the Control Repository 232
Enabling Monolithic and Per-module Hybrid Deployment 232
Moving Modules to their Own Repositories 233
Placing Roles and Profiles in the site/ Module Directory 235
Remove Fully Qualified Paths 236
Moving Shared Tools to Their Own Repository 237
Implementing Test Cases 237
Best Practices for Puppet Deployments 237
Using Repository Access Control to Enforce Deployment Policy 238
Enabling Multiteam Coordination 239
Pinning Module Versions 241

Isolating Puppet Extensions 242
Utilizing Standard Environment Configuration Practices 243
Git Best Practices 245
Deployment Practices 245
Summary 246

10. Extending Puppet. 249
The Cost of Extending Puppet 249
Minimizing Development Costs 249
Reducing Upgrade Costs 250
Testing 250
Static Code Analysis 251
Unit Testing 252
Acceptance Testing 255
Creating Facts 257
Distributing Facts in Modules 258
Facts Puppet Can't Know 259
Structured Facts 259
Abusing Facts 260
Trusted Certificate Attributes 260
Custom Types and Providers 260
Avoiding Creation of Duplicate Types 261
Creating a New Resource Type 263
Custom Resource Providers 265
Reuse Existing Frameworks 267
Creating Custom Hiera Backends 268
Choose the Appropriate Backend Type 269
Creating a High-Performance Backend 269
Using Puppet's Public Classes and Method Calls 270
Puppet Faces 271
Indirection 271
Deploying Extensions 272
pluginsync 272
Deploying Gem Dependencies 272
Deploying Ruby Gem Extensions on Puppet Server 273
Summary 273

Index. 275