Valsts IKT risināju izdarbes (DevOps) vadlīnijas

Viedās administrācijas un reģionālās attīstības ministrija

Rīga 2026

Versija	v.1.0.
Datums	20.04.2026

Izmaiņu vēsture

Versija	Datums	Izmaiņas	Autors
v.1.0	20.04.2026	Papildināts un pilnveidots saskaņā ar saņemtajiem priekšlikumiem.	U. Karlovs-Karlovskis.

Saturs

Saīsinājumi un Termini

Saīsinājums/termins	Skaidrojums
API	Lietojumprogrammu saskarne (ang. val. *Application Programming Interface*) - definētu noteikumu un metožu kopums, kas nodrošina standartizētu datu un funkcionalitātes apmaiņu starp informācijas sistēmām vai programmatūras komponentēm.
CI	Nepārtrauktā integrācija (ang. val. *Continuous Integration*) - programmatūras izstrādes prakse, kurā izmaiņas pirmkodā un konfigurācijās tiek regulāri integrētas centralizētā repozitorijā un automātiski pārbaudītas.
CI/CD	Nepārtrauktā integrācija un nepārtrauktā piegāde (ang. val. *Continuous Integration / Continuous Delivery*) - automatizēta pieeja programmatūras izmaiņu integrēšanai, pārbaudei un sagatavošanai producijas vidē.
IaC	Infrastruktūra kā kods (ang. val. *Infrastructure as Code*) ir prakse, kurā infrastruktūras resursi tiek pārvaldīti un nodrošināti, izmantojot kodu, nevis manuālas procesuālas darbības.
Minimālais vienībtestu pārklājums	Minimālais vienībtestu pārklājums tiek noteikts projekta vai produkta līmenī, ņemot vērā risinājuma kritiskumu, riska līmeni un tehnoloģisko kontekstu. Ja esošajam (vēsturiskajam) kodam vienībtestu pārklājums nav nodrošināts, prasība primāri attiecināma uz jaunu vai modificētu kodu, vienlaikus pakāpeniski uzlabojot kopējo testu pārklājumu sistēmas attīstības gaitā.
VaFCeRP	Valsts Federatīvi Centralizēto Repozitoriju Platforma ir vienota, pārvaldīta platforma publiski finansētas programmatūras pirmkoda, konfigurāciju, tehniskās dokumentācijas un ar programmatūras darbināšanu saistīto artefaktu glabāšanai, versiju kontrolei un izmaiņu plūsmas pārvaldībai.
Laidiena kandidāts	Laidiena kandidāts ir CI procesa rezultātā izveidots, testēts un validēts artefakts, kas ir gatavs izvietošanai vidēs bez papildu izmaiņām pirmkodā un konfigurācijās un ir izgājis visas definētās kvalitātes un drošības pārbaudes.

1 Ievads

1.1 Vadlīniju mērķis

Vadlīniju mērķis ir noteikt vienotu pieeju programmatūras izstrādei, uzturēšanai un darbināšanai valsts pārvaldē, balstoties uz vienotu Valsts Federatīvi Centralizētu Repozitoriju Platformu (turpmāk - VaFCeRP) un saskaņotām izdarbes (ang. val. *DevOps*) praksēm. Tas nodrošina, ka publiski finansētas programmatūras pirmkods un dokumentācija tiek pārvaldīta caurspīdīgi, droši un atkārtoti izmantojami. Vadlīnijas veicina programmatūras dzīves cikla pārvaldību atbilstoši industrijas labajai praksei, integrējot automatizāciju, kvalitātes kontroli un drošības pārbaudi.

1.2 Vadlīniju tvērums

Vadlīnijas nosaka vienotu ietvaru visām prasībām un principiem, kas regulē programmatūras izstrādi, izvietošanu, uzturēšanu un pārvaldību valsts pārvaldē. Tās ir piemērojamas visam programmatūras dzīves ciklam un aptver gan tehniskos, gan organizatoriskos aspektus, nodrošinot saskaņotu izdarbes pieeju īstenošanu. Vadlīnijas attiecas uz visu valsts pārvaldi un tās projektiem neatkarīgi no izmantojamām tehnoloģijām un piegādātāja.

Vadlīniju tvērumā ietilpst arī gadījumi, kuros programmatūras, platformu vai infrastruktūras risinājumu izmantošanas ietvaros tiek veikta konfigurācija, automatizācija vai pielāgošana, izmantojot kodu, tostarp konfigurācija kā kods.

Šādas darbības ir uzskatāmas par programmatūras izstrādes sastāvdaļu neatkarīgi no tā, vai pamatrisinājums ir balstīts uz programmproduktu īres (ang. val. *SaaS)*, platformas kā pakalpojuma (ang. val. *PaaS)* vai infrastruktūras kā pakalpojuma (ang. val. *IaaS)*[1].

Ja informācijas sistēma atbilstoši normatīvajiem aktiem ir klasificēta kā kritiskā informācijas sistēma vai tai piemērojamas paaugstinātas drošības prasības[2] (piemēram, A kategorijas sistēmas), šo vadlīniju piemērošana īstenojama, ievērojot attiecīgo drošības iestāžu un kompetento institūciju noteiktos ierobežojumus un prasības. Šādos gadījumos publiskojamā pirmkoda un ar to saistītās informācijas apjoms var tikt ierobežots, pamatojoties uz drošības prasībām, tomēr ierobežojumiem jābūt dokumentētiem, pamatotiem un samērīgiem ar aizsargājamo informāciju.

1.3 Ārpus tvēruma

Vadlīnijas neattiecas uz informācijas sistēmām un risinājumiem, kuru ieviešanas vai izmantošanas ietvaros netiek veikta programmatūras pirmkoda izstrāde, modificēšana vai uzturēšana valsts pārvaldes pusē.

[1] Valsts pārvaldes rīcībā esošo datu apstrādei nepieciešamas specializētas lietojumprogrammatūras sagādes modeļi

[2] https://likumi.lv/ta/id/361481-minimalas-kiberdrosibas-prasibas

Ārpus šo vadlīniju tvēruma ir, tostarp, bet ne tikai:

- programmatūras kā pakalpojuma (ang. val. *SaaS*) risinājumi, kuru izmantošana aprobežojas tikai ar standarta funkcionalitātes lietošanu, neveicot pielāgojumus, skriptu izstrādi, konfigurācijas izmaiņas koda veidā vai citu izstrādes darbību;
- komerciāli gatavi programmatūras produkti (ang. val. *COTS*), kas tiek izmantoti bez būtiskām pielāgošanām vai koda izmaiņām;
- mākoņpakalpojumi un platformas, kuru izmantošana aprobežojas ar konfigurēšanu, nevis programmatūras izstrādi;
- ārējo pakalpojumu sniedzēju pārvaldītas lietotnes, kurās valsts pārvalde neveic izstrādes, testēšanas vai izvietošanas darbības.

Šādos gadījumos piemērojamas citas normatīvās prasības un vadlīnijas, kas regulē iepirkumus, pakalpojumu pārvaldību, informācijas drošību un datu aizsardzību, bet ne programmatūras izstrādes dzīves cikla pārvaldību šī dokumenta izpratnē.

1.4 Saistītie dokumenti

Apkopoti normatīvie akti, politikas plānošanas dokumenti un vadlīnijas, kas pamato šo izdarbes vadlīniju un valsts federatīvā programmatūras repozitorija izveides mērķi un nepieciešamību. Dokumenti ir grupēti pēc to satura tvēruma un attiecības pret praktisku izpildes instrukciju nepieciešamību.

1.4.1. Dokumenti, kas nosaka nepieciešamību pēc vienotām vadlīnijām praktiskai īstenošanai

MK noteikumi "Informācijas sistēmu vispārējās tehniskās prasības"[3] nosaka vispārējās tehniskās prasības informācijas sistēmām, t. sk. institūciju pienākumu tās ievērot kā vienotu pamatu izstrādei/uzturēšanai.

Valsts informācijas sistēmu likums[4] nosaka mērķi un uzdevumus - vienotu kārtību, kā valsts/institūciju IS veido, reģistrē, uztur, koplieto u. c.

Digitālās transformācijas pamatnostādnes 2021–2027[5] nosaka, ka valsts pārvaldes IS tiek radītas primāri balstoties uz atvērtā koda risinājumiem

Nacionālās kiberdrošības likums[6] paredz, ka Ministru kabinets nosaka minimālās kiberdrošības prasības un kārtību, kā subjekti nodrošina tīklu, IS atbilstību, t. sk. konfidencialitāti, integritāti, pieejamību un datu atjaunošanu.

[3] https://likumi.lv/ta/id/343549-informacijas-sistemu-visparejas-tehniskas-prasibas

[4] https://likumi.lv/ta/id/62324-valsts-informacijas-sistemu-likums

[5] https://likumi.lv/ta/id/324715-par-digitalas-transformacijas-pamatnostadnem-20212027-gadam

[6] https://likumi.lv/ta/id/353390-nacionalas-kiberdrosibas-likums

Digitālās pārvaldes arhitektūras ietvarā horizontālo jomu arhitektūra[7] nosaka koplietošanas risinājumus, platformas, komponentes un pakalpojumus, kurus izmanto visās valsts pārvaldes nozarēs. Viena no horizontālajām jomām ir IKT infrastruktūra un kiberdrošība, uz kuras risinājumiem un pakalpojumiem tiek balstīti pārējo jomu risinājumi. Šīs vadlīnijas ir izstrādātas atbilstoši minētajai arhitektūras pieejai un konkretizē principus programmatūras izstrādei, repozitoriju izmantošanai, konfigurācijas pārvaldībai un izdarbes prakšu piemērošanai valsts pārvaldē.

1.4.2. Dokumenti, kuros ir noteikta attīstības, izmaiņu un pārvaldības kārtība, bet iztrūkst izstrādes un piegādes praktisko noteikumu

MK noteikumi par VIRSIS[8] nosaka vienotu valsts līmeņa informācijas sistēmu, IKT resursu un pakalpojumu uzskaites un pārvaldības kārtību, t. sk. reģistrēšanas un informācijas aprites principus, kas veicina pārskatāmību un standartizāciju valsts pārvaldē, tomēr tie nenosaka vienotu pirmkoda pārvaldības un piegādes izpildes modeli, piem., repozitorija struktūru, zarošanu, koda pārskatīšanu un CI/CD prasības.

MK noteikumi Nr. 368[9] nosaka informācijas sistēmu un to darbībai nepieciešamo IKT resursu/pakalpojumu attīstības aktivitāšu un likvidēšanas uzraudzības kārtību, institucionalizējot attīstības aktivitāšu reģistrāciju un saskaņošanu valsts līmenī, taču praktiskie izpildes noteikumi programmatūras izstrādē un piegādē šajā regulējumā nav detalizēti definēti un ir nostiprināmi izdarbes vadlīnijās.

Valsts datu apstrādes mākoņa noteikumi (MK Nr. 606)[10] nosaka valsts mākoņdatošanas platformu un mākoņpakalpojumu pārvaldības ietvaru, lomas un atbildību sadalījumu, kā arī pakalpojumu izmantošanas kārtību, tostarp ievērojot kiberdrošības prasības, tomēr tie neparedz vienotu standartu kopumu programmatūras izstrādes darba plūsmai un piegādes automatizācijai, tādēļ šie aspekti jānosaka izdarbes vadlīnijās.

2 Programmatūras izvietošanas un izmantošanas nosacījumi

Programmatūras izvietošana un izmantošana valsts pārvaldē jāīsteno saskaņā ar standartizētiem izdarbes principiem, nodrošinot drošu, atkārtojamu un automatizētu programmatūras dzīves ciklu. Visiem izstrādātajiem un izvietotajiem programmatūras risinājumiem, kuru izstrāde tiek finansēta no publiskajiem līdzekļiem, jābūt publiski pieejamiem, tostarp to pirmkodam un saistītajai tehniskajai dokumentācijai, izmantojot VaFCeRP, ja vien tas nav pretrunā ar normatīvajiem aktiem, tostarp Nacionālās kiberdrošības likumā noteiktajiem ierobežojumiem.[6] Programmatūras

[7] https://www.varam.gov.lv/lv/horizontalo-jomu-arhitekturas

[8] https://likumi.lv/ta/id/349664-valsts-informacijas-resursu-sistemu-un-sadarbspejas-informacijas-sistemas-noteikumi

[9] https://likumi.lv/ta/id/343550-informacijas-sistemu-un-to-darbibai-nepieciesamo-informacijas-un-komunikacijas-tehnologiju-resursu-un-pakalpojumu-attistibas-aktivitasu-un-likvidesanas-uzraudzibas-kartiba

[10] https://likumi.lv/ta/id/363717-valsts-datu-apstrades-makona-noteikumi

risinājumiem jābūt ilgtspējīgiem, caurspīdīgi auditējamiem un neatkarīgiem no konkrēta izstrādātāja vai piegādātāja.

Programmatūras izstrādes un izvietošanas procesā ir jānodrošina:

- Versiju kontrole VaFCeRP;
- Automatizēta būvēšana, testēšana un izvietošana;
- Drošības un kvalitātes pārbaude visos programmatūras dzīves cikla posmos.

Programmatūras izvietošanas ietvaros programmatūras būvēšanas rezultātā izveidotās pakotnes un citi izvietošanai paredzētie artefakti ir jāpublicē VaFCeRP sistēmā. Repozitorija sistēmai jānodrošina publicēto pakotņu un artefaktu lejupielādes iespējas programmatūras izvietošanai un izmantošanai dažādās vidēs. Izvietošanas process jāorganizē tā, lai nodrošinātu izsekojamību starp programmatūras pirmkodu, būvēšanas rezultātiem un izmantotajām pakotnēm.

2.1 Licences

Valsts pārvaldē izstrādātās programmatūras pirmkodam jābūt publiski pieejamam un izplatītam saskaņā ar atvērtā koda licenci, kas nodrošina tiesības programmatūru izmantot, pētīt, modificēt un pārizplatīt, ievērojot licences nosacījumus. Publiski finansētas programmatūras izstrādē ir jāievēro princips "Public Money, Public Code"[11], nodrošinot pirmkoda publisku pieejamību un atkārtotu izmantojamību kā atvērtā pirmkoda risinājumam.

Izvēloties informācijas tehnoloģiju risinājumus, prioritāri ir jāizvērtē un, ja iespējams, jāizmanto atvērtā pirmkoda programmatūra ("Open Source First"[12] princips), mazinot piegādātāju atkarību un veicinot risinājumu ilgtspēju un pārnesamību. Iespēju robežās priekšroka dodama Eiropā izstrādātiem digitālajiem risinājumiem ("European Preference"[13] princips), stiprinot Eiropas Savienības digitālo suverenitāti un neatkarību.

Izvēlētajai licencei jānodrošina:

- juridiska skaidrība par programmatūras izmantošanas un pārizmantošanas tiesībām;
- saderība ar publiskās pārvaldes mērķiem un atvērtas pārvaldības principiem;
- iespēja programmatūru atkārtoti izmantot citos valsts pārvaldes projektos un sistēmās.

Atkarībā no publicējamā satura veida ieteicams izmantot piemērotas licences, piemēram:

- European Union Public Licence (EUPL);
- Open Data Commons Open Database License (ODbL);

[11] https://publiccode.eu/en/

[12] https://commission.europa.eu/about/departments-and-executive-agencies/digital-services/open-source-software-strategy_en

[13] https://ec.europa.eu/isa2/eif_en/

- Creative Commons Attribution (CC BY);
- Creative Commons Zero Public Domain Dedication (CC0);

2.2 Pārizmantojamība

Programmatūra ir jāizstrādā ar mērķi nodrošināt tās maksimālu pārizmantojamību citos projektos vai sistēmās. Pārizmantojamība ir jāīsteno kā arhitektūras princips nevis kā papildus funkcionalitāte.

Pārizmantojamības ieviešanai tiek īstenota ar sekojošiem nosacījumiem:

- Programmatūra ir jābūvē modulāri, ar nodalītām komponentēm;
- Jāizmanto standartizēti API un atvērtie protokoli;
- Konfigurācijas un vides mainīgie jāatdala no pirmkoda;
- Jānodrošina pilna tehniskā dokumentācija (instalācija, konfigurācija, izmantošana);
- Jāizmanto standartizēti un vispārpieņemti rīki.

Pārizmantojamai programmatūrai jābūt:

- Viegli izvietojamai dažādās infrastruktūrās;
- Pielāgojamai bez papildu koda izmaiņām;
- Viegli uzturamai neatkarīgi no izstrādātāja.

Lai veicinātu programmatūras praktisku pārizmantojamību, ieteicams ievērot šādus principus:

- atkārtoti izmantojamās programmatūras komponentes noformēt kā atsevišķas, loģiski nodalītas programmatūras vienības ar skaidri definētām saskarnēm, kas ļauj tās izmantot citos risinājumos bez manuālām pielāgošanām;
- nodrošināt automatizētu laidienu sagatavošanu un publicēšanu, kas garantē izmantoto komponentu izcelsmes, versiju un savstarpējo atkarību izsekojamību;
- nodrošināt augsta līmeņa dokumentāciju, kas skaidro risinājuma mērķi, paredzētos izmantošanas scenārijus un integrācijas principus, kā arī iekšēju dokumentāciju kodā;
- nodrošināt, ka būtiskā programmatūras funkcionalitāte ir pārbaudīta ar automatizētiem vienības testiem, kas tiek izpildīti standarta būvēšanas procesa ietvaros.

3 Konfigurācijas pārvaldības vadlīnijas

Konfigurācijas pārvaldība nodrošina programmatūras risinājumu stabilu, drošu un atkārtojamu izstrādi, izvietošanu un darbināšanu visā programmatūras dzīves ciklā. Tās mērķis ir nodrošināt, ka visas ar programmatūras darbību saistītās vienības ir identificējamas, kontrolējamas un izsekojamas, samazinot manuālu darbību un kļūdu risku.

Visas konfigurācijas un ar tām saistītās vienības ir jāpārvalda kā kods un jāglabā VaFCeRP sistēmā, nodrošinot pilnīgu izmaiņu vēsturi un auditējamību.

3.1 Artefakti un konfigurācijas vienības

Konfigurācijas vienība ir jebkura pakalpojuma komponente, infrastruktūras elements, artefakts vai cita vienība, kuru nepieciešams pārvaldīt, lai nodrošinātu sekmīgu pakalpojuma vai programmatūras risinājuma izstrādi, uzturēšanu un darbināšanu.

Konfigurācijas vienības ietver, tostarp, bet ne tikai:

- programmatūras pirmkodu un tā komponentes;
- programmatūras bibliotēkas un atkārtoti izmantojamus komponentus;
- MI modeļus un saistītās datu kopas;
- konfigurācijas, parametrus un vides definīcijas;
- automatizācijas un izvietošanas definīcijas;
- infrastruktūras un platformu konfigurācijas aprakstus;
- ar programmatūras darbību saistītos artefaktus.

Programmatūras bibliotēkas un atkārtoti izmantojamie komponenti ir uzskatāmi par atsevišķām konfigurācijas vienībām neatkarīgi no to izcelsmes vai izmantošanas veida.

Konfigurācijas vienībām jābūt:

- identificējamām un unikāli nosakāmām;
- pakļautām vienotai un kontrolētai versiju pārvaldībai;
- izsekojamām visā programmatūras dzīves ciklā;
- glabātām centralizēti;
- reproducējamām dažādās vidēs bez izmaiņām pašās vienībās.

Katra konfigurācijas vienība ir jāapraksta ar konfigurācijas pārvaldībai nepieciešamo informāciju, kas nodrošina pilnīgu izsekojamību un auditējamību.

Konfigurācijas vienības aprakstā jāietver vismaz:

- unikāls identifikators;
- versiju vai versiju kopa;
- konfigurācijas vienības izveides un izmaiņu vēsture;
- informācija par izmaiņu autoru un apstiprinātāju;
- izmaiņu apraksts un pamatojums;
- attiecības ar citām konfigurācijas vienībām un programmatūras laidieniem.

Izmaiņas konfigurācijas vienībās, tostarp bibliotēku pievienošana, aizstāšana vai versiju maiņa, ir uzskatāmas par konfigurācijas izmaiņām un jāpakļauj izmaiņu pārvaldības procesam.

3.2 Versiju pārvaldība

Konfigurācijas vienībām jābūt pakļautām vienotai versiju pārvaldībai visā to dzīves ciklā. Versiju pārvaldība nodrošina iespēju reproducēt sistēmas stāvokli noteiktā laika brīdī un atjaunot iepriekšējas konfigurācijas kļūmju vai incidentu gadījumā.
Versiju pārvaldības ietvaros ir jānodrošina:

- skaidra un konsekventa konfigurācijas vienību versiju piešķiršana;
- konfigurācijas versiju sasaistīšana ar programmatūras laidieniem;
- iespēja identificēt konkrētā vidē izmantotās konfigurācijas vienības un to versijas;
- droša atgriešanās pie iepriekšējām konfigurācijas versijām.

3.3 Versiju kontrole

Visām konfigurācijas vienībām jābūt glabātām VaFCeRP sistēmā. Versiju kontrole nodrošina izmaiņu caurspīdīgumu, atbildības sadalījumu un izsekojamību.
Versiju kontroles ietvaros jāievēro šādi principi:

- Konfigurācijas vienībām jābūt uzturētām versiju kontroles sistēmā neatkarīgi no izmaiņu veikšanas veida;
- manuālas izmaiņas mērķa vidēs ārpus noteiktās izmaiņu pārvaldības kārtības nav pieļaujamas;
- katrai izmaiņai jābūt dokumentētai ar izmaiņu aprakstu;
- jānodrošina piekļuves tiesību kontrole;
- jāizmanto strukturēta izmaiņu ieviešanas plūsma ar pārskatīšanu un apstiprināšanu.

3.4 Izmaiņu pārvaldība

Visas konfigurācijas izmaiņas jāpakļauj formālam izmaiņu pārvaldības procesam. Izmaiņu pārvaldība nodrošina sistēmas stabilitāti, drošību un paredzamu darbību.
Izmaiņu pārvaldības procesā ir jānodrošina:

- izmaiņu ietekmes izvērtēšana pirms to ieviešanas;
- izmaiņu apstiprināšana atbilstoši noteiktajām lomām;
- izmaiņu testēšana pirms izvietošanas;
- izmaiņu izsekojamība visā to dzīves ciklā;
- iespēja ātri atjaunot iepriekšējās konfigurācijas vērtības un to stāvokli.

3.5 Validācija

Visām konfigurācijas vienībām un konfigurācijām pirms to izmantošanas jāveic validācija. Validācijas mērķis ir nodrošināt atbilstību noteiktajām prasībām, drošības principiem un savietojamību ar mērķa vidi.

Validācijas ietvaros jānodrošina:

- konfigurāciju struktūras un loģikas pārbaude;
- neatbilstošu vai nedrošu iestatījumu identificēšana;

- savietojamības pārbaude starp konfigurācijas vienībām;
- neatbilstošu konfigurāciju bloķēšana no izvietošanas.

Validācijas rezultāti ir jāfiksē validācijas žurnālos un jānodrošina to pieejamība pārbaudes un uzraudzības nolūkos. Validācija var tikt veikta gan automatizēti, gan manuāli, savukārt automatizēta validācijas integrācija būvēšanas un izvietošanas procesos ir ieteicama kā labā prakse, lai nodrošinātu konsekventu un atkārtojamu konfigurāciju pārbaudi pirms to izmantošanas.

3.6 Zarošanās stratēģijas un to pielietojums

Zarošanas stratēģija versiju kontroles sistēmā ir būtiska konfigurācijas pārvaldības sastāvdaļa, kas nodrošina kontrolētu izmaiņu ieviešanu, paralēlu izstrādi un stabilu programmatūras laidienu pārvaldību. Pareizi definēta zarošanas pieeja samazina riskus, uzlabo izsekojamību un nodrošina konfigurāciju konsekvenci visā programmatūras dzīves ciklā.

Zarošanas stratēģijai jānodrošina, ka konfigurācijas vienību izmaiņas tiek veiktas strukturēti, pārskatāmi un saskaņoti ar izmaiņu un laidienu pārvaldības procesiem.

Vispārīgie principi

Zarošanas stratēģijas ietvaros jāievēro šādi principi:

- katrai būtiskai izmaiņai konfigurācijā vai programmatūras funkcionalitātē jābūt izolētai atsevišķā zarā;
- galvenajā zarā drīkst atrasties tikai pārskatītas, apstiprinātas un testētas konfigurācijas;
- zaru nosaukumiem un lietojumam jābūt vienoti definētiem un saprotamiem visām iesaistītajām pusēm;
- konfigurācijas izmaiņām dažādās vidēs jābūt izsekojamām caur zaru struktūru.

Lai nodrošinātu kontrolētu, pārskatāmu un konsekventu konfigurācijas un programmatūras izmaiņu pārvaldību, ir ieteicams izmantot strukturētu zarošanas modeli ar skaidri definētiem zaru veidiem. Šāda pieeja palīdz saglabāt galvenā zara stabilitāti, atbalsta paralēlu izstrādi un nodrošina izmaiņu izsekojamību visā programmatūras dzīves ciklā.

Aprakstīto zaru veidu ievērošana ir uzskatāma par labu praksi konfigurācijas, laidienu un izmaiņu pārvaldībā.

Galvenais zars

Galvenais zars (*ang. val. main/master/trunk*) kalpo kā autoritatīvais konfigurāciju un programmatūras stāvokļa avots. Tas satur:

- stabilas un apstiprinātas konfigurācijas vienības;
- izmaiņas, kas ir izturējušas noteikto izmaiņu pārvaldības, testēšanas un validācijas procesu;
- konfigurācijas, kas atbilst vai ir sagatavotas nākamajam programmatūras laidienam.

Tiešas izmaiņas galvenajā zarā nav pieļaujamas. Visas izmaiņas jāievieš caur strukturētu pārskatīšanas un apstiprināšanas plūsmu.

Funkcionalitātes un izmaiņu zari
Funkcionalitātes vai izmaiņu zari tiek veidoti konkrētu izmaiņu, uzlabojumu vai kļūdu labojumu ieviešanai. Šajos zaros:

- tiek veiktas konfigurācijas un koda izmaiņas, kas attiecas uz konkrētu uzdevumu;
- izmaiņas var tikt attīstītas, testētas un koriģētas neatkarīgi no galvenā zara;
- pēc pabeigšanas izmaiņas tiek iesniegtas pārskatīšanai un apstiprināšanai pirms apvienošanas ar galveno zaru.

Laidienu zari
Laidienu zari tiek veidoti no galvenā zara brīdī, kad tiek gatavots programmatūras laidiena kandidāts. Laidienu zaru mērķis ir:

- fiksēt konkrētam laidienam paredzēto konfigurāciju kopu;
- veikt tikai ar laidiena stabilizāciju saistītas izmaiņas, piemēram, kļūdu labojumus vai dokumentācijas precizējumus;
- nodrošināt iespēju atkārtoti izveidot un izvietot konkrēta laidiena konfigurācijas.

Laidienu zari kalpo kā atsauce konkrētām programmatūras versijām un nodrošina pilnīgu konfigurācijas reproducējamību.

Karsto labojumu zari
Kritisku incidentu vai drošības ievainojamību gadījumā var tikt izmantoti karsto labojumu zari. Šie zari:

- tiek veidoti no galvenā vai laidiena zara;
- satur tikai minimāli nepieciešamās izmaiņas problēmas novēršanai;
- pēc ieviešanas tiek apvienoti atpakaļ attiecīgajos zaros, lai saglabātu konfigurāciju konsekvenci.

Zarošanas stratēģijai jābūt cieši integrētai ar konfigurācijas pārvaldības procesiem, nodrošinot, ka:

1. katra konfigurācijas vienības versija ir sasaistāma ar konkrētu zaru un programmatūras laidienu;
2. izmaiņu vēsture ir pilnībā izsekojama;
3. ir iespējams ātri atjaunot iepriekšējos konfigurācijas stāvokļus;
4. konfigurācijas atbilst definētajām drošības, stabilitātes un validācijas prasībām.

3.7 MLOps un MI modeļu pārvaldība

Mākslīgā intelekta (MI) un mašīnmācīšanās izdarbe (kopā ang. val. akronīms *MLOps*) ir organizējama atbilstoši šajās vadlīnijās noteiktajiem konfigurācijas pārvaldības, versiju kontroles

un izsekojamības principiem, vienlaikus ievērojot ar datu izmantošanu un modeļu apmācību saistīto specifiku.

Papildus programmatūras pirmkodam un infrastruktūrai par konfigurācijas vienībām ir uzskatāmi arī:

- MI modeļi un to versijas;
- modeļu apmācības, validācijas un testēšanas datu kopas vai atsauces uz tām;
- datu priekšapstrādes un transformācijas loģika;
- modeļu apmācības parametri un konfigurācijas;
- uzvednes (ang. val. *prompts*), tostarp sistēmas uzvednes (ang. val. *system prompts*).

MI risinājumu izstrādes un uzturēšanas ietvaros ir jānodrošina:

- datu izcelsmes (ang. val. *data lineage*) izsekojamība, nodrošinot iespēju identificēt izmantotās datu kopas un to versijas;
- modeļu reproducējamība, nodrošinot iespēju atkārtoti iegūt identisku rezultātu, izmantojot to pašu konfigurāciju un datu kopu;
- visu ar modeli saistīto konfigurācijas vienību versiju pārvaldība;
- automatizēta modeļu validācija un kvalitātes novērtēšana pirms izvietošanas.

Uzvednes un citi MI modeļa darbību ietekmējoši konfigurācijas elementi ir uzskatāmi par konfigurācijas vienībām, un attiecībā uz tiem ir jānodrošina:

- versiju pārvaldība un glabāšana repozitorijā;
- pakļaušana izmaiņu pārvaldības un pārskatīšanas procesam;
- sasaistāmība ar konkrētām modeļa versijām un izvietojumiem.

MI risinājumu izvietošanā ir jānodrošina izsekojamība starp izmantoto modeļa versiju, datu kopām, konfigurāciju un izvietojumu konkrētā vidē. Izdarbes procesi ir integrējami kopējā nepārtrauktās integrācijas un piegādes (CI/CD) pieejā, nodrošinot automatizētu modeļu būvēšanu, testēšanu, validāciju un izvietošanu.

4 Nepārtrauktā integrācija

Nepārtrauktās integrācijas ieviešana nodrošina, ka programmatūras izmaiņas tiek apstrādātas centralizēti, pārskatāmi un atkārtojami, izmantojot automatizētus procesus. Tas ļauj samazināt manuālu darbību risku, savlaicīgi identificēt neatbilstības un nodrošināt atbilstību drošības, kvalitātes un arhitektūras prasībām visā programmatūras dzīves ciklā.

4.1 Valsts federatīvi centralizēto repozitoriju platformas definīcija un loma

VaFCeRP ir vienota, pārvaldīta platforma, kas nodrošina publiski finansētas programmatūras pirmkoda, konfigurāciju, tehniskās dokumentācijas un ar programmatūras darbināšanu saistīto artefaktu glabāšanu, versiju kontroli un izmaiņu plūsmas pārvaldību. Valsts repozitorijam ir jānodrošina integrēta nepārtrauktās integrācijas funkcionalitāte, tostarp automatizēta būvēšana, testēšana un pārbaužu izpilde, kā arī artefaktu publicēšana.

Esošie valsts iestāžu vai privātie repozitoriji netiek aizstāti vai likvidēti, tie var turpināt darboties savās vidēs, taču tiem ir jānodrošina sinhronizācija ar VaFCeRP, lai nodrošinātu publiski finansētas programmatūras pirmkoda, artefaktu un ar tiem saistītās informācijas pieejamību un pārvaldību vienotā valsts līmeņa platformā.

Repozitorijs kalpo kā autoritatīvais avots programmatūras izmaiņu vēsturei un laidienu reproducējamībai, nodrošinot izsekojamību starp pirmkodu, būvēšanas rezultātiem, izmantotajām pakotnēm un izvietojumiem. Repozitorija sistēmai jānodrošina piekļuves tiesību kontrole un darbību uzskaite, lai nodrošinātu auditējamību un atbildības sadalījumu.

Valsts repozitoriju var izmantot valsts un pašvaldību iestādes, kā arī izstrādātāji un piegādātāji, kas izstrādā programmatūru publiskajam sektoram vai vēlas publicēt pirmkodu ar atvērtā pirmkoda licenci - EUPL.

4.2 Repozitorijā izvietojamais saturs

Centrālajā repozitorijā ir jāizvieto un jāuztur saturs, kas ir nepieciešams programmatūras izstrādei, uzturēšanai, reproducējamai būvēšanai un izvietošanai. Repozitorijā izvietojamajam saturam jānodrošina iespēja identificēt un atjaunot sistēmas stāvokli noteiktā laika brīdī, kā arī izsekot izmaiņas visā dzīves ciklā.

Centrālajā repozitorijā ir jāuztur vismaz:

- programmatūras pirmkods, tā komponentes, bibliotēkas un atkārtoti izmantojami komponenti;
- konfigurācijas, parametri un vides definīcijas, tostarp konfigurācija kā kods;
- automatizācijas definīcijas (būvēšana, testēšana, izvietošana) un to izpildes skripti;
- kvalitātes un drošības pārbaužu definīcijas, ja tās ir attiecināmas;
- tehniskā dokumentācija, tostarp uzstādīšanas, konfigurēšanas un darbināšanas apraksti;
- izmaiņu žurnāli un laidienu informācija;
- būvēšanas rezultātā izveidotie artefakti, publicēti centralizēti un sasaistīti ar pirmkoda versiju;
- ar katru laidiena kandidātu saistītais programmatūras pirmkods.

Atbilstoši industrijas labajai praksei, kopā ar laidiena kandidātu jānodod arī pilnu koda izmaiņu vēsturi, lai nodrošinātu caurspīdīgu un auditējamu izstrādes procesu no pirmā koda ieraksta līdz gatavam artefaktam. Šāda pieeja garantē pilnīgu izsekojamību starp izstrādes gaitu un piegādāto risinājumu, kā arī nodrošina laidienu precīzu reproducējamību.

Repozitorijā nedrīkst glabāt noslēpumus (ang. val. *Secrets*[14]) atklātā veidā. Sensitīva informācija, ja tā ir nepieciešama izstrādes vai darbināšanas nolūkiem, ir jāapstrādā atbilstoši informācijas drošības un datu aizsardzības prasībām.

[14] https://cheatsheetseries.owasp.org/cheatsheets/Secrets_Management_Cheat_Sheet.html

4.3 Atbildības, lomu sadalījums, Informācijas piekļuve

Repozitorija un nepārtrauktās integrācijas procesu pārvaldība ir jāorganizē tā, lai nodrošinātu skaidru atbildību sadalījumu, kontrolētu piekļuvi un izmaiņu izsekojamību. Informācijas sistēmas attīstības ietvaros informācijas sistēmas pārzinis nodrošina vispārējo tehnisko prasību ievērošanu un to izpildi. Nepārtrauktās integrācijas īstenošana valsts pārvaldē ir balstāma uz centralizētu programmatūras dzīves cikla nodrošināšanas platformu, kas atbalsta izstrāddarbināšanas principus.

Lomas un atbildības[15]

Lomu sadalījumā piemērojamas šādas lomas, nodrošinot savstarpēju pienākumu nodalījumu un kontroles mehānismus:

- **Valsts informācijas sistēmas pārzinis**
  Institūcija, kas normatīvajos aktos noteiktajā kārtībā organizē un vada valsts informācijas sistēmas darbību. Pārzinis nodrošina informācijas sistēmas attīstības, izmaiņu ieviešanas un uzturēšanas procesu pārvaldību, kā arī to, ka izmaiņas tiek ieviestas kontrolētā, izsekojamā un auditējamā veidā.
- **Pārziņa pilnvarotā persona**
  Persona vai struktūrvienība, kas pārziņa noteiktajā pilnvarojuma apjomā veic ar informācijas sistēmas uzturēšanu, repozitorija pārvaldību vai CI/CD procesa darbību saistītus operacionālos uzdevumus, tostarp piekļuves tiesību administrēšanu un tehnisko darbību izpildi.
- **Kiberdrošības pārvaldnieks**
  Persona vai struktūrvienība, kas nodrošina kiberdrošības prasību ievērošanas kontroli attiecīgajā tvērumā, tai skaitā uzrauga noteiktu drošības pasākumu izpildi un atbilstību piemērojamiem normatīvajiem aktiem.
- **Izmaiņu ieviesējs (izstrādātājs/uzturētājs)**
  Persona vai piegādātāja pārstāvis, kas sagatavo izmaiņas pirmkodā, konfigurācijās un automatizācijās, nodrošina atbilstību noteiktajai izmaiņu ieviešanas plūsmai, kā arī sagatavo izmaiņas pārskatīšanai un integrācijai.

4.4 Nepārtrauktās integrācijas principi

Nepārtrauktās integrācijas (ang. val. *Continuous Integration* jeb *CI*) procesā ir jānodrošina automatizētas pārbaudes, kas tiek izpildītas pirms izmaiņu integrācijas galvenajā zarā un pirms laidienu kandidāta (ang. val. *Release Candidate*) sagatavošanas. Pārbaužu apjomam un stingrībai jāatbilst risinājuma kritiskumam un riska līmenim.

Nepārtrauktās integrācijas ietvaros ir jānodrošina vismaz:

- automatizēta koda būvēšana un laidiena kandidāta izveide ar plūsma-kā-kods (ang. val. *pipeline as code*) principu pēc izmaiņu iesniegšanas repozitorijā;

[15] https://likumi.lv/ta/id/62324-valsts-informacijas-sistemu-likums

- automatizēta vienībtestēšana kā daļa no būvēšanas procesa;
- uzstādīts minimālais pieļaujamais vienībtestu pārklājums kodam;
- konfigurāciju un automatizāciju validācija, ja tā ir attiecināma uz konkrēto risinājumu;
- automatizētas koda kvalitātes un drošības pārbaudes atbilstoši risinājuma kritiskumam un riska līmenim;
- būvēšanas rezultātu izsekojamība, nodrošinot sasaisti starp pirmkoda versiju, būvi un publicētajiem artefaktiem;
- CI izpildes rezultātu saglabāšana un pieejamība pārskatīšanai.

CI izpildes rezultāti ir jāsaglabā un jānodrošina pieejamība pārskatīšanas un uzraudzības nolūkos. CI definīcijām un izmaiņām automatizācijas plūsmā jābūt uzturētām kā kodam un pakļautām tai pašai pārskatīšanas un apstiprināšanas kārtībai kā programmatūras pirmkoda izmaiņām.

5 Nepārtraukta piegāde

Nepārtraukta piegāde (ang. val. *Continuous Delivery* jeb *CD*) ir programmatūras dzīves cikla posms, kura primārais mērķis ir nodrošināt, ka pēc sekmīgas nepārtrauktās integrācijas izveidotais laidiena kandidāts tiek kontrolēti, droši un izsekojami nogādāts līdz gala lietotājam vai produkcijas videi.

Nepārtrauktas piegādes uzdevums nav tikai tehniska izvietošana, bet strukturēta laidiena sagatavošana, validēšana un virzīšana caur definētiem starpposmiem līdz brīdim, kad tas ir gatavs lietošanai produkcijā. Šos starpposmus definē atbildīgie par projektu atbilstoši labajai praksei, piemēram, ISO/IEC 20000-1:2018. Šis process nodrošina saikni starp pirmkoda versiju, būvēšanas rezultātu (artefaktu) un izvietojumu konkrētā vidē, tādējādi garantējot pilnīgu izsekojamību un reproducējamību.

5.1 CI/CD un automatizācijas atbalsts

Piegādes un izvietošanas īstenošana jābalsta uz automatizētām nepārtrauktās integrācijas un nepārtrauktās piegādes plūsmām (ang. val. *CI/CD pipelines*), kas definētas kā kods un uzturētas centralizētā repozitorijā. CI/CD plūsmām jābūt versijpārvaldītām, pārskatāmām un apstiprināmām tāpat kā programmatūras pirmkodam.

Piegādes ietvaros ir jānošķir:

- **Nepārtraukta piegāde** (ang. val. *Continuous Delivery*) **-** process, kurā programmatūras izmaiņas tiek automatizēti sagatavotas izvietošanai produkcijas vidē, bet faktiska izvietošana ir pakļauta vismaz vienam manuālam apstiprinājumam;

- **Nepārtraukta izvietošana** (ang. val. *Continuous Deployment*) **-** process, kurā programmatūras izmaiņas pēc visu pārbaužu sekmīgas izpildes tiek automātiski izvietotas produkcijas vidē bez papildu manuālas iesaistes.

Nepārtraukta izvietošana ir ieteicama, bet to drīkst piemērot tikai tad, ja ir nodrošināts pietiekams automatizēto testu pārklājums, drošības pārbaužu integrācija un piegādes risku izvērtējums atbilstoši sistēmas kritiskumam. Augsta kritiskuma informācijas sistēmās nepārtrauktas izvietošanas princips nav pieļaujams un ir jālieto nepārtraukta piegāde.

Neatkarīgi no principa, visām CI/CD plūsmām jānodrošina:

- automatizēta programmatūras būvēšana un artefaktu sagatavošana;
- kvalitātes kontroles mehānismi pirms pārejas uz nākamo vidi;
- strukturēta pāreja starp vidēm;
- izvietošanas procesa izsekojamība;
- izvietošanas rezultātu reģistrēšana un saglabāšana.

Ieteicams veidot pārizmantojamas automatizācijas plūsmas un moduļus, lai nodrošinātu vienotu pieeju dažādos projektos un samazinātu paralēlu un atšķirīgu risinājumu veidošanu. Automatizācijas plūsmām jābūt standartizētām un centralizēti uzturētām.

Pirms pārejas uz nākamo vidi (t. i., nākamo definēto izvietošanas posmu CI/CD plūsmā, skat. 5.2.) jāpiemēro kvalitātes kontroles slieksnis (ang. val. *quality gate*), kas nodrošina, ka ir izpildītas attiecīgajam posmam definētās kvalitātes prasības. Atkarībā no plūsmas struktūras šādi posmi var ietvert būvēšanu, testēšanu, pakotņu sagatavošanu, drošības pārbaudes vai izvietošanu.

Kvalitātes kontroles slieksnim ir jānodrošina:

- veiksmīgi pabeigtu būvēšanas procesu;
- izpildītus vienībtestus ar sasniegtu minimālo pārklājuma līmeni;
- drošības pārbaudes bez kritiskānm ievainojamībām;
- konfigurācijas validāciju;
- atbilstību arhitektūras un infrastruktūras prasībām.

Lai mazinātu darbības pārtraukuma risku, produkcijas un pirmsprodukcijas vidēs ieteicams izmantot pakāpeniskas izvietošanas stratēģijas, piemēram, zilās-zaļās (ang. val. *Blue-Green*[16]) izvietošanas pieeju vai kanārijputniņa (ang. val. *Canary*) [17]pieeju ar pakāpenisku lietotāju plūsmas novirzīšanu. Izvietošanas procesam jāparedz iespēja veikt kontrolētu atgriešanos pie iepriekšējās versijas.

5.2 Programmatūras izvietošana vidē/s

Programmatūras izvietošana jāīsteno automatizēti, izmantojot nepārtrauktās integrācijas un nepārtrauktās piegādes plūsmas. Artefaktu nedrīkst mainīt vai pārbūvēt pārejā starp vidēm, lai nodrošinātu konsekvenci un samazinātu neatbilstību risku. Ja pastāv objektīva nepieciešamība

[16] https://www.redhat.com/en/topics/devops/what-is-blue-green-deployment

[17] https://docs.gitlab.com/user/project/canary_deployments/

veikt atkārtotu būvēšanu vai pielāgojumus, tiem jābūt dokumentētiem, pamatotiem un izsekojamiem.

Katram izvietojumam jābūt identificējamam un sasaistāmam ar konkrētu pirmkoda versiju, būves identifikatoru un attiecīgo konfigurācijas versiju. Jānodrošina iespēja izsekot izmaiņu izcelsmei un nepieciešamības gadījumā atjaunot iepriekšējo stāvokli.

Manuālas izmaiņas mērķa vidēs ārpus definētā procesa nav pieļaujamas. Visas vides jādefinē un jāpārvalda kā infrastruktūra kā kods (*ang. val. Infrastructure as Code, IaC*).

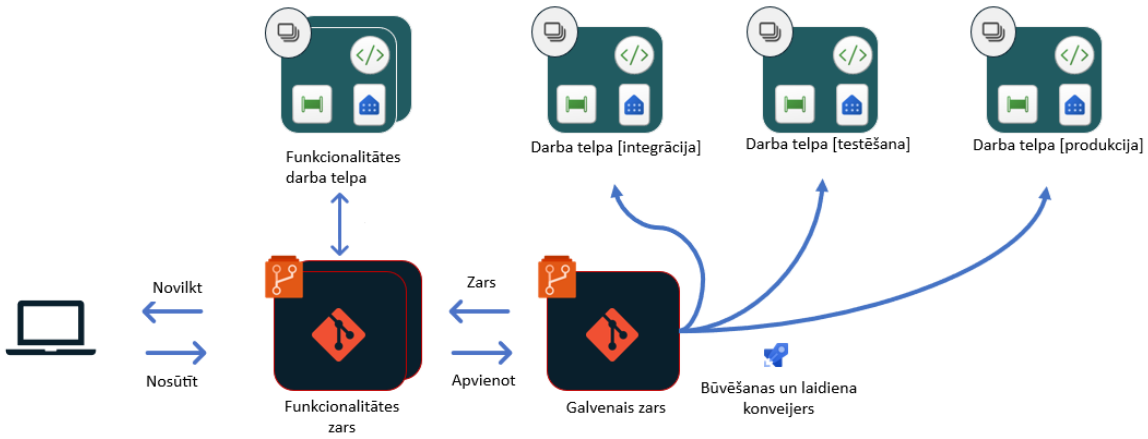
Programmatūras izvietošanā ieteicams izmantot konteinerizāciju[18], nodrošinot lietojumprogrammas un tās atkarību iepakošanu izolētā, reproducējamā izpildes vienībā, artefaktā. Konteineru izmantošana veicina konsekvenci starp vidēm, samazina konfigurācijas neatbilstību risku un atvieglo automatizētu izvietošanu, mērogošanu un versiju pārvaldību. Lietot konteinerizāciju visās vidēs tiek uzskatīta par labās prakses pieeju.

Programmatūras dzīves ciklā parasti tiek izmantotas šādas vides:

1) **Lokālā izstrādes vide -** Paredzēta individuālai izstrādei un sākotnējai validācijai. Nodrošina iespēju reproducēt sistēmas darbību kontrolētā, izstrādātājam pieejamā vidē.
2) **Integrācijas vide -** Koplietojama vide, kur tiek integrētas vairāku izstrādātāju izmaiņas. Šajā vidē automātiski tiek izvietoti artefakti pēc veiksmīgas būves un kvalitātes kontroles posma.
3) **Testēšanas vide -** Paredzēta funkcionālai, regresijas un nefunkcionālai testēšanai. Nodrošina sistēmas atbilstības validāciju pirms pārejas uz nākamo vidi.
4) **Pirmsprodukcijas vide (***ang. val. staging***) -** Pēc iespējas identiska produkcijas videi. Šī vide primāri paredzēta izmantošanai pakāpeniskās izvietošanas scenārijos, piemēram, Zilās-Zaļās izvietošanas pieejā. Kamēr tā netiek izmantota pārslēgšanai, ieteicams to izmantot automatizētu testu izpildei, sistēmas akcepttestēšanai un lietotāju apmācībām.
5) **Produkcijas vide -** Sistēmas faktiskā darbināšanas vide. Izvietošana jāveic kontrolēti, ar pilnu izsekojamību un iespēju veikt atgriešanos pie iepriekšējās versijas.

Sākot ar testēšanas vidi un visos turpmākajos posmos, visām programmatūras atkarībām, tostarp tranzitīvajām atkarībām jābūt ar fiksētām versijām. Nav pieļaujama mainīgu vai nenoteiktu versiju izmantošana, kas var radīt neatkārtojamību starp vidēm vai uz laika līnijas. Atbildība par izmantoto atkarību izvērtēšanu un atbilstību saskaņā ar kiberdrošības un citiem noteikumiem gulstas uz izmaiņu autoru un uzturētāju.

[18] https://opencontainers.org/



Tulkots no avota: (https://learn.microsoft.com/en-us/fabric/cicd/manage-deployment)

Infrastruktūra jāapraksta kā kods, nodrošinot:

- infrastruktūras elementu versiju pārvaldību;
- izmaiņu izsekojamību;
- iespēju reproducēt vidi no nulles;
- konsekventu drošības un piekļuves politiku ieviešanu.

IaC ir uzskatāma par konfigurācijas vienību un ir pakļaujama tādai pašai versiju kontroles, izmaiņu pārvaldības un validācijas kārtībai kā programmatūras pirmkods.

5.3. Infrastruktūras kā koda organizēšanas principi

Infrastruktūra kā kods jāorganizē tā, lai nodrošinātu pārizmantojamību, standartizāciju un skaidru atbildību sadalījumu. IaC struktūrai jāatbilst sistēmas arhitektūrai un jāatbalsta izvēlētais izstrādes un darbināšanas modelis.

IaC ietvaros jāparedz koplietojami moduļi un vienota koda bāze, kas nodrošina infrastruktūras elementu konsekventu definēšanu. Koplietojamie moduļi var ietvert, piemēram, tīkla konfigurācijas, drošības politikas, identitātes pārvaldības komponentes un citus atkārtoti izmantojamus infrastruktūras elementus.

Viengabalainas (monolīta) arhitektūras gadījumā infrastruktūra kā kods var tikt uzturēt tajā pašā repozitorijā, kurā tiek uzturēts lietojumprogrammas pirmkods, ja tas nodrošina uzturēšanas vienkāršību un nepārkāpj drošības prasības. Šādā gadījumā IaC tiek uzturēts kā integrēta koda bāzes daļa, saglabājot vienotu versiju kontroli un izmaiņu izsekojamību.

Mikroservisu arhitektūras gadījumā jāizvērtē atsevišķu repozitoriju izmantošana katrai lietojumprogrammai vai pakalpojumam VaFCeRP. Šādā modelī koplietojamie infrastruktūras moduļi var tikt uzturēti centralizēti, savukārt katrs pakalpojums savā repozitorijā izmanto šos

moduļus atbilstoši savām vajadzībām. Šāda pieeja nodrošina komandu autonomiju, vienlaikus saglabājot standartizāciju infrastruktūras līmenī.

Neatkarīgi no izvēlētās struktūras jānodrošina, ka:

- koplietojamie IaC moduļiem ir nodrošināta versiju kontrole un tie ir dokumentēti;
- infrastruktūras definīcijas ir auditējamas un reproducējamas;
- atkarības starp lietojumprogrammu IaC un koplietojamajiem moduļiem ir skaidri definētas;
- drošības un piekļuves politikas tiek ieviestas konsekventi visā platformā.

IaC struktūra jādefinē projekta sākumposmā un jānostiprina kā saistoša izstrādes un darbināšanas prakse, nodrošinot pārvaldāmu un ilgtspējīgu infrastruktūras attīstību.

6 Standartizācija un tehnoloģiskā uzskaite

Standartizācija un tehnoloģiskā uzskaite nodrošina vienotu pieeju tehnoloģiju izmantošanai, pārvaldībai un attīstībai valsts pārvaldē. Tās mērķis ir samazināt fragmentāciju starp risinājumiem, veicināt pārizmantojamību, nodrošināt drošības prasību ievērošanu un uzlabot programmatūras risinājumu ilgtspējīgu uzturēšanu. Tehnoloģiju izvērtēšanā un katalogu uzturēšanā ieteicams ievērot Eiropas Savienības savietojamības principus, European Interoperability Framework (EIF)[19] un Interoperable Europe Act[20] vadlīnijas, kas veicina publiskā sektora programmatūras savietojamību, atkārtotu izmantošanu un tehnoloģisko neitralitāti.

Standartizācijas pieeja nenosaka obligātu vienas tehnoloģijas izmantošanu visos gadījumos, bet nosaka vienotus principus tehnoloģiju izvērtēšanai, dokumentēšanai un uzskaitei. Tas nodrošina, ka tehnoloģiskā vide ir pārskatāma, auditējama un ilgtspējīgi uzturama.

Tehnoloģiskā uzskaite jāveic centralizēti un jāatjauno visā programmatūras dzīves ciklā, nodrošinot pilnīgu informāciju par izmantotajām tehnoloģijām, rīkiem un platformām.

6.1 Rīku un tehnoloģiju uzskaite

Programmatūras risinājumu izstrāde un darbināšana balstās uz dažādu tehnoloģiju, rīku un platformu kopumu, kas veido sistēmas tehnoloģisko vidi (ang. val. *technology stack*). Lai nodrošinātu pārskatāmību par izmantotajām tehnoloģijām, visām informācijas sistēmām jānodrošina to sistemātiska uzskaite.

Tehnoloģiju uzskaite ļauj identificēt sistēmā izmantotās komponentes, to versijas un savstarpējās atkarības, kā arī savlaicīgi izvērtēt drošības riskus un uzturēšanas nepieciešamību.

[19] https://interoperable-europe.ec.europa.eu/collection/iopeu-monitoring/1-introduction-european-interoperability-framework

[20] https://commission.europa.eu/publications/interoperable-europe-act_en

Tehnoloģiju uzskaitei jābūt uzturētai visā programmatūras dzīves ciklā un jāatjauno ikreiz, kad regulāras inventarizācijas rezultātā tiek identificētas nepilnības, sistēmā tiek ieviestas jaunas tehnoloģijas vai veiktas būtiskas izmaiņas esošajās komponentēs.

Izvēloties tehnoloģijas, ieteicams dot priekšroku plaši izmantotām un starptautiski atzītām atvērtā koda tehnoloģijām, kurām ir aktīva izstrādātāju kopiena, regulāri drošības atjauninājumi un pieejama dokumentācija. Šādu tehnoloģiju izmantošana samazina piegādātāja atkarības risku (ang. val. *vendor lock-in*) un uzlabo risinājumu pārnesamību starp infrastruktūrām.

Par starptautiski atzītām un plaši izmantotām atvērtā koda tehnoloģijām var uzskatīt, piemēram, šādas tehnoloģijas:

- Konteinerizācija un orķestrācija:
  - Docker
  - Kubernetes
- CI/CD un automatizācija:
  - GitLab CI/CD
  - Jenkins
  - Argo CD
- Infrastruktūra kā kods:
  - Terraform
  - Ansible
- Uzraudzība:
  - Prometheus
  - Grafana
  - OpenSearch / Elasticsearch
  - Zabbix
  - Loki

6.2 Standarta tehnoloģiju katalogs

Lai nodrošinātu pārskatāmību par izmantotajām tehnoloģijām un veicinātu pārizmantojamību valsts pārvaldē, katrai informācijas sistēmai ir jāpārizmanto esošais vai jāuztur savs tehnoloģiju katalogs. Šādā katalogā tiek apkopotas konkrētajā informācijas sistēmā izmantotās tehnoloģijas, rīki un platformas, kas nepieciešamas sistēmas izstrādei, darbināšanai un uzturēšanai.

Standarta tehnoloģiju katalogs nodrošina pārskatāmu informāciju par sistēmas tehnoloģisko vidi un ļauj identificēt izmantotās komponentes, to versijas un savstarpējās atkarības. Tas palīdz plānot sistēmas uzturēšanu, izvērtēt drošības riskus un pieņemt pamatotus lēmumus par sistēmas turpmāko attīstību.

Tehnoloģiju katalogā ir jāiekļauj, bet ne tikai:

- programmēšanas valodas un izstrādes ietvarus;
- datubāzu pārvaldības sistēmas;
- konteinerizācijas un orķestrācijas platformas;

- nepārtrauktās integrācijas un piegādes rīkus;
- infrastruktūras pārvaldības rīkus;
- sistēmu novērojamības un darbības uzraudzības rīkus;
- drošības testēšanas un koda kvalitātes pārbaudes rīkus;
- citus saistītos rīkus un komponentes.

Katalogā katrai tehnoloģijai ir jānorāda vismaz:

- tehnoloģijas nosaukumu;
  rekomendēto versiju vai versiju diapazonu;
- izmantošanas jomu;
- statusu (piemēram: rekomendēta, pieļaujama, novecojoša);
- plānotās izmaiņas, ja nepieciešams;
- papildu piezīmes vai ierobežojumus.

Tehnoloģiju katalogs jāaktualizē, veicot būtiskas izmaiņas sistēmas tehnoloģiskajā vidē, kā arī regulāri pārskatot izmantotās tehnoloģijas un to versijas.

Ja informācijas sistēmas ir vairāk par vienu, papildus ieteicams uzturēt centralizētu pārskatu par informācijas sistēmu tehnoloģiju katalogiem. Šādā kopkatalogā tiek reģistrētas visas informācijas sistēmas un to tehnoloģiju katalogi, norādot pēdējās tehnoloģiju inventarizācijas datumu un atbildīgo institūciju vai uzturētāju. Centralizēta pārskata uzturēšana ļauj identificēt koplietošanas iespējas starp sistēmām, uzraudzīt tehnoloģiju izmantošanas tendences un savlaicīgi konstatēt novecojošas vai drošības riskus radošas tehnoloģijas.

Ja sistēmā tiek izmantotas trešo pušu bibliotēkas vai komponentes, jānodrošina to ievainojamību uzraudzība un versiju pārvaldība. Ieteicams izmantot automatizētus rīkus atkarību analīzei un drošības ievainojamību identificēšanai.

Dokumenta piemērošana un turpmākā attīstība

Vadlīnijas ir izstrādātas, lai atbalstītu valsts IKT infrastruktūras un kiberdrošības arhitektūras[21] mērķu un rezultātu sasniegšanu, kas ir saskaņoti ar valsts politiku un mērķiem digitālās transformācijas jomā, nosakot vienotus principus un pieeju programmatūras izstrādei, pārvaldībai un darbināšanai. Šīs vadlīnijas ir saistītas ar valsts IKT infrastruktūras un kiberdrošības arhitektūru, kuras ieviešanai tiks īstenots ERAF 2021.-2027. projekts "IKT infrastruktūras un kiberdrošības arhitektūras ieviešana (1. kārta)". Minētā projekta ietvaros tiks izstrādāta VaFCeRP platforma, kurai ir jādarbojas atbilstoši šajās vadlīnijās noteiktajiem principiem. Vadlīnijas ir piemērojamas visām valsts pārvaldes institūcijām un projektiem, kuri izstrādā, uztur vai attīsta programmatūru publiskā finansējuma ietvaros, kā arī izmanto vai integrējas ar VaFCeRP.

[21] Horizontālo jomu arhitektūras | Viedās administrācijas un reģionālās attīstības ministrija

Vadlīnijas nosaka vienotu izpratni par VaFCeRP darbības principiem, nodrošinot strukturētu, drošu un pārskatāmu programmatūras dzīves cikla pārvaldību, kā arī veicinot atvērtā koda izmantošanu, pārizmantojamību un standartizētu izstrādes praksi valsts pārvaldē.

Šīs vadlīnijas tiek noteiktas kā obligāti piemērojams dokuments, nodrošinot vienotu pieeju programmatūras izstrādei un pārvaldībai valsts pārvaldē. Vadlīnijas kalpo kā praktisks pamats vienotai izpratnei par platformas darbības principiem, veicinot efektīvu sadarbību, kvalitātes un drošības prasību ievērošanu, kā arī ilgtspējīgu un koordinētu valsts IKT risinājumu attīstību.